Duty-Cycled Wireless Sensor Networks: Wakeup Scheduling, Routing, and Broadcasting

Shouwen Lai

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Binoy Ravindran, Chair
Paul Plassmann
Y. Thomas Hou
Anil Vullikanti
Yaling Yang

April 26, 2010
Blacksburg, Virginia

Keywords: Wireless Sensor Network, Duty Cycle, MAC, Routing, Broadcast, Quorum System

Duty-Cycled Wireless Sensor Networks: Wakeup Scheduling, Routing and Broadcasting

Shouwen Lai

(ABSTRACT)

In order to save energy consumption in idle states, low duty-cycled operation is widely used in Wireless Sensor Networks (WSNs), where each node periodically switches between sleeping mode and awake mode. Although efficient toward saving energy, duty-cycling causes many challenges, such as difficulty in neighbor discovery due to asynchronous wakeup/sleep scheduling, time-varying transmission latencies due to varying neighbor discovery latencies, and difficulty on multihop broadcasting due to non-simultaneous wakeup in neighborhood. This dissertation focuses on this problem space. Specifically, we focus on three co-related problems in duty-cycled WSNs: wakeup scheduling, routing and broadcasting.

We propose an asynchronous quorum-based wakeup scheduling scheme, which optimizes heterogenous energy saving ratio and achieves bounded neighbor discovery latency, without requiring time synchronization. Our solution is based on quorum system design. We propose two designs: *cyclic quorum system pair* (*cqs-pair*) and *grid quorum system pair* (*gqs-pair*). We also present fast offline construction algorithms for such designs. Our analytical and experimental results show that cqs-pair and gqs-pair achieve better trade-off between the average discovery delay and energy consumption ratio. We also study asymmetric quorum-based wakeup scheduling for two-tiered network topologies for further improving energy efficiency.

Heterogenous duty-cycling causes transmission latencies to be time-varying. Hence, the routing problem becomes more complex when the time domain must be considered for data delivery in duty-cycled WSNs. We formulate the routing problem as time-dependent Bellman-Ford problem, and use vector representation for time-varying link costs and end-to-end (E2E) distances. We present efficient algorithms for route construction and maintenance, which have bounded time and message complexities in the worst case by ameliorating with β-synchronizer.

Multihop broadcast is complex in duty-cycled WSNs due to non simultaneous wakeup in neighborhoods. We present Hybrid-cast, an asynchronous multihop broadcast protocol, which can be applied to low duty-cycling or quorum-based duty-cycling schedules, where nodes send out a beacon message at the beginning of wakeup slots. Hybrid-cast achieves better tradeoff between broadcast latency and broadcast count compared to previous broadcast solutions. It adopts opportunistic data delivery in order to reduce the broadcast latency. Meanwhile, it reduces redundant transmission via delivery deferring and online forwarder selection. We analytically establish the upper bound of broadcast count and the broadcast latency under Hybrid-cast.

To verify the feasibility, effectiveness, and performance of our solutions for asynchronous wakeup scheduling, we developed a prototype implementation using Telosb and TinyOS 2.0 WSN platforms. We integrated our algorithms with the existing protocol stack in TinyOS, and compared them with the CSMA mechanism. Our implementation measurements illustrate the feasibility, performance trade-off, and effectiveness of the proposed solutions for low duty-cycled WSNs.

This work was supported by the Ministry of Knowledge Economy (MKE) of the Republic of Korea. [2008-F-052, Scalable/Mobile/Reliable Wireless Sensor Network Technology].

To my parents and my wife (Jinyan)

感谢父母，含辛茹苦，育我成长
感谢妻子，相濡以沫，陪伴身旁

Acknowledgments

After staying at Virginia Tech for three years, I am pleased to have reached this step. In the past years, many times I felt frustrated and pressured. I am lucky to get through all those difficulties with the help and support from many people, and I would like to thank them.

First of all, I would like to thank my advisor, Dr. Binoy Ravindran. He offered me the chance of working with him from my first day at Virginia Tech. Thanks for his advice and support in the initial stage of my PhD when I was weak to start research in the Real-time Systems Laboratory. I learned a lot from him on academic writing, presentations, and selecting correct research directions. He always encouraged me to do research of the highest quality even when I suffered set-backs at first. I also highly appreciate his faith in me, allowing me to select my favorite research directions in these years.

Thanks, too, to the rest of my committee: Dr. Tom Hou, Dr. Paul Plassmann, Dr. Anil Vullikanti, and Dr. Yaling Yang. I greatly appreciate their support and encouragement on my work, and also their willingness to bend their schedules to accommodate the meetings and exams. I am also grateful to Dr. Hyeonjoong Cho, who is an alumni of my laboratory and now works at Korea University. He offered me many suggestions and comments when I prepared my first conference paper. I also thank him for securing the financial support from ETRI, Korea, which supported my dissertation research.

In addition, I would like to thank all my colleagues in the Real-Time Systems Laboratory who discussed with me many valuable ideas of my research. They include Kai Han, Jonathan Anderson, Bo Jiang, Bo Zhang, Guanhong Pei, Fei Huang, Sherif Fahmy, Piyush Garyali, and Junwhan Kim. I am also grateful to all my friends at VT for helping me in innumerable ways, especially for those serving with me in VT-ACSS, GSA Research Symposium, and BSO Budget Board.

Lastly, I would like to thank my family members for all their love and support. I am grateful to my parents for their hard work which allowed me to afford my education from when I was a child to when I became a college student. Without their love, I would not have reached this milestone and this dissertation would not have happened. I would like to thank especially my elder brother who is always a role model for me, inspiring me as I was growing up. Finally, I am forever indebted to my wife Jinyan Fu. She has been incredibly supportive of me through the ups and downs over the past years. I cannot thank her enough for being such a wonderful wife.

This dissertation is to all people who have helped and are helping me.

Contents

List of Figures

List of Tables

Nomenclature

ALPL adaptive low power listing, page 11

ATIM Ad hoc Traffic Indication Map, page 97

cqs-pair cyclic quorum system-pair, page 6

CSMA Carrier Sense Multiple Access, page 4

E2E End-to-End, page 7

FIFO First-In-First-Out, page 59

FTSP fast time-dependent shortest path, page 67

FTSP-M fast maintenance algorithm for time-dependent shortest path, page 70

gqs-pair grid quorum system pair, page iii

h-QPS heterogeneous quorum-based power saving, page 22

LCM least common multiple , page 75

LPL low power listening, page 3

MCDS minimum connecting dominating set, page 87

MPR multipoint relays, page 88

p-Grid prime grid quorum system, page 7

QPS quorum-based power saving, page 4

S-MAC Sensor-MAC, page 4

TDSP time-dependent shortest path, page 59

WSN Wireless Sensor Networks, page 2

Chapter 1

Introduction

Wireless Sensor Networks (WSN) consist of spatially distributed autonomous sensor nodes which are organized into a cooperative network [2]. WSNs are usually deployed to monitor physical or environmental properties, such as temperature, vibration, pressure, motion, or pollutants. The development of WSNs was initially motivated by military applications such as battlefield surveillance. However, they are increasingly being used in many industrial and civilian application domains, including industrial process monitoring and control [3], machine health monitoring [4], environment and habitat monitoring [5], and medical diagnostics [6].

In WSNs, each node consists of a micro-processor, multiple types of memory (program, data and flash memories), RF transceiver, various sensors and actuators, and power supplies (e.g., batteries and solar cells). A WSN normally constitutes a wireless ad-hoc network, meaning that each sensor node supports a multihop routing algorithm, and several nodes may forward data packets to a base station via a sink node. A typical multihop architecture for WSNs is shown in Figure 1.1.

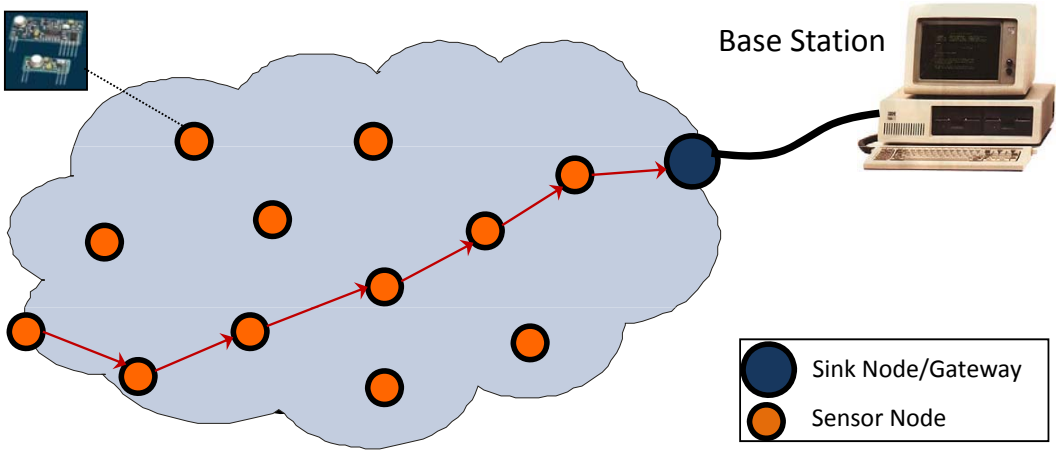


Figure 1.1: Typical Multihop WSN Architecture

A sensor node is usually constrained by small size and low cost [7]. Such constraints affect the design of power supplies, memory, computational speed, and bandwidth. In particular, sensor nodes usually rely on portable power sources such as batteries to provide the necessary power. Thus, energy efficiency is critically important for WSNs [2].

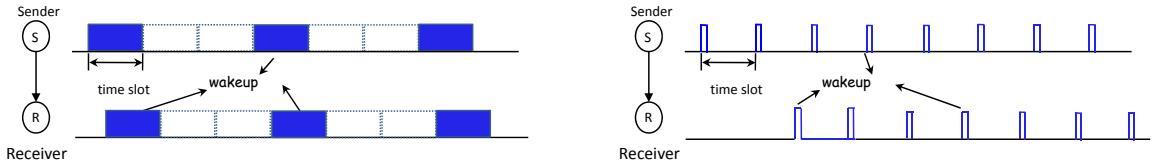
1.1 Duty-cycled Wireless Sensor Networks

It has been observed that idle energy plays an important role for saving energy in WSNs [8]. Most existing radios (i.e., CC2420 [9]) used in WSNs support different modes, such as transmit/receive mode, idle mode, and sleep mode. In the idle mode, the radio is not communicating but the radio circuitry is still turned on, resulting in energy consumption which is only slightly less than that in the transmitting or receiving states. Thus, a better way is to shut down the radio as much as possible in the idle mode [8]. The typical energy consumption parameters for a Telosb [10] are shown in Table 1.1

Table 1.1: Energy consumption of different components in Telosb

Module	Power	Remarks
Processor/memory	1.8 mA	Active mode
Processor/memory	5.1 μA	Sleep mode
Radio RX mode	18.8 mA	receiving
Radio TX mode	17.4 mA	transmission
Radio Idle mode	21 μA	
Radio Sleep mode	1 μA	

Suppose time is arranged into consecutive and equal time slots. Now, two modes for low duty cycle operation can be identified: slotted listening mode [11, 12] and low power listening mode [13]. In the slotted listening mode, as shown in Figure 1.2(a), a node is wholly awake in select slots and asleep in the remaining slots when there is no data transmission or reception. In the low power listening (LPL) mode, as shown in Figure 1.2(b), a node will be fractionally awake in every slot.



(a) Slotted listening (b) Low power listening

Figure 1.2: Duty-cycled operation in WSNs

We define duty cycle as the percentage of time a node is active in the whole operational time. Generally, the duty cycle in the LPL mode is lower than that in the slotted listening mode.

Adaptive duty-cycling has been proposed in the recent works on energy-harvesting technologies [14, 15], such as solar power [16], to replenish battery supply in WSNs. Due to high costs and the unavailability of a continuous power supply, it is not feasible to have instantly sufficient energy output. Hence, saving idle energy consumption is still necessary. Adaptive duty cycling [15, 17] is thus proposed to save energy consumption and to prolong the sustainable workable time per node. The duty cycle setting can be based on the residual energy [18], node location, or the rechargeable energy [17] on each node, independently.

Although low and adaptively duty-cycled operations can yield greater energy efficiency for WSNs, neighbor discovery becomes more complex than that in conventional works for always-on mechanisms (e.g., CSMA), since we cannot guarantee that two nodes are awake simultaneously.

1.2 Problem Spaces and Motivations

To support duty cycling, it is necessary to introduce a wakeup scheduling scheme in which a node sleeps in more slots in idle state, but maintains network connectivity. Towards this goal, existing neighbor discovery mechanisms fall into three categories: on-demand wakeup, scheduled neighbor discovery, and asynchronous neighbor discovery.

In on-demand wakeup mechanisms [19–22], out-of-band signaling or operational cycle is used to wake up sleeping nodes in an on-demand manner. For example, with the help of a paging signal, a node listening on a page channel can be awakened. As page radios can operate at lower power consumption, this strategy is very energy efficient. However, it suffers from increased implementation complexity.

In scheduled wakeup mechanisms [23–25], low-power sleeping nodes wake up at the same time, periodically, to communicate with one another. Examples include the S-MAC protocol [23] and the multi-parent schemes protocol [26]. In such schemes, all nodes maintain periodic sleep-listen schedules based on locally managed synchronization. Neighboring nodes form virtual clusters to set up a common sleep schedule.

The third category, asynchronous wakeup mechanisms [12, 13, 27–29] are also well studied. Compared to the scheduled neighbor discovery wakeup mechanism, asynchronous wakeup does not require clock synchronization. In this approach, each node follows its own wakeup schedule in the idle state, as long as the wakeup intervals among neighbors overlap. To meet this requirement, nodes usually have to wakeup more frequently than in the scheduled neighbor discovery mechanism. However, there are many advantages of asynchronous wakeup, such as easiness in implementation and low message overhead for communication. Furthermore, it can ensure network connectivity even in highly dynamic networks.

The quorum-based wakeup scheduling paradigm, sometimes called quorum-based power saving (QPS) protocol [11, 30–32], is an asynchronous wakeup mechanism in slotted listening mode, and has been proposed as a powerful solution for asynchronous wakeup scheduling. In a QPS

protocol, the time axis on each node is evenly divided into beacon intervals. Given an integer n, a quorum system defines a cycle pattern, which specifies the awake/sleep scheduling pattern during n continuous beacon intervals for each node. We call n the *cycle length*, since the pattern repeats every n beacon intervals. A node may stay awake or sleep during each beacon interval. QPS protocols guarantee that at least one awake interval overlaps between two adjacent nodes, with each node being awake for only $O(\sqrt{n})$ beacon intervals.

Most previous works only consider *homogenous* quorum systems for asynchronous wakeup scheduling [12], which means that quorum systems for all nodes have the same *cycle length* and same pattern. However, many WSNs are increasingly heterogenous in nature—i.e., the network nodes are grouped into clusters, with each cluster having a high-power cluster head node and low-power cluster member nodes [33–38].

Thus, the first problem that we focus in this dissertation is heterogenous quorum systems design, where heterogenous sensor nodes (i.e., clusterheads and cluster members) have different quorum-based wakeup schedules (or different *cycle lengths*). We denote two quorums from different quorum systems as *heterogenous quorums* in this dissertation. If two adjacent nodes adopt heterogenous quorums as their wakeup schedules, they have different cycle lengths and different wakeup patterns. Hence, how to guarantee the intersection property for heterogenous quorums and apply them for wakeup scheduling in WSNs with slotted listening mode is a non-trivial problem.

Secondly, it is desirable to have an asymmetric design in which there are two types of quorums: *read* and *write*. The read quorum has a smaller quorum size compared to the write quorum. If we can guarantee that a read quorum and a write quorum have the non-empty intersection property (i.e., $write \cap write \neq \emptyset$, $read \cap write \neq \emptyset$, but not necessarily $read \cap read \neq \emptyset$), then we can apply *read* quorum to cluster members and *write* quorum to cluster head, because there is usually no communication between cluster members. If the read quorums have smaller quorum size, it will achieve higher energy saving ratio comparing with that of write quorums.

Although quorum-based wakeup scheduling has been shown to achieve excellent idle energy savings, scalability, and easiness in implementation, they suffer from time-varying neighbor discovery latencies, which is also pointed out by Ye *et.al.* [39]. As shown in Figure 5.1, the neighbor discovery latency between two neighbors is varying at different departure times. Even with synchronized duty cycling, the neighbor discovery latency is varying at different time moments due to adaptive duty cycle setting as shown in Figure 5.1. Thus, this raises another fundamental problem: with time-varying link costs, how to find optimal paths with least nodes-to-sink latency for all nodes at all discrete departure time moments? This is a non-trivial problem, and is the second problem that we study in the dissertation.

The last problem that we consider is multihop broadcasting [40], which is an important network service in WSNs, especially for applications such as code update, remote network configuration, route discovery, etc. Although the problem of broadcast has been well studied in always-on networks [41, 42], such as wireless ad hoc networks where neighbor connectivity is not a problem, broadcast is more difficult in duty-cycled WSNs where each node stays awake only for a fraction of the time slots and neighborhood nodes are not simultaneously awake for receiving data. The

problem becomes more difficult in asynchronous [12] and heterogenous duty-cycling [43] scenarios.

The dissertation focuses on the three problems described above. We solve all three problems with efficient protocol and algorithm design, that are efficient in terms of latency bound, energy efficiency, and run-time complexity. We will also analyze the performance of our solutions theoretically, and verify their effectiveness and efficiency by simulation studies and prototype implementations.

1.3 Summary of Research Contributions

The goals to address the three problems in the dissertation include: 1. To maintain network connectivity in duty-cycled WSNs. Here, we use the term "connectivity" loosely, in the sense that a topologically connected network in our context may not be connected at any given time; instead, all nodes are reachable from a node within a finite amount of time. 2. To design a fast distributed algorithm for the time-varying shortest path routing problem, which can efficiently enumerate all optimal paths with least end-to-sink latencies for infinite time intervals. In addition, design an algorithm which can dynamically and distributively maintain time-dependent least-latency paths. 3. To broadcast messages asynchronously to entire network in a multihop manner, with low flooding latency and low message cost.

Toward these goals, we have developed a set of solutions. We conclude our contributions as follows.

- cqs-pair [1]. We developed the cyclic quorum system pair (or cqs-pair), which guarantees that two adjacent nodes which adopt heterogenous cyclic quorums from such a pair as their wakeup schedules, can hear each other at least once within one super cycle length (i.e., the larger cycle length in the cqs-pair).

  We also developed a fast algorithm for constructing cqs-pairs, using the *multiplier theorem* [44] and the (N, k, M, l)-difference pair defined by us. Given a pair of cycle lengths (n and m, $n \leq m$), we show that the cqs-pair is an optimal design in terms of energy saving ratio. Our fast construction scheme significantly improves the speed of finding an optimal quorum, in contrast to previous exhaustive methods [45]. We also analyze the performance of cqs-pair in terms of expected delay ($\frac{n-1}{2} < E(delay) < \frac{m-1}{2}$), quorum ratio, and energy saving ratio.

  With the help of the cqs-pair, WSNs can achieve better trade-off between energy consumption and average delay. For example, all cluster-heads and gateway nodes can select a quorum from the quorum system with smaller cycle length as their wake up schedules, to obtain smaller discovery delay. In addition, all members in a cluster can choose a quorum from the quorum system with longer cycle length as their wakeup schedules, in order to save more idle energy. To the best of our knowledge, cqs-pair is the first solution that can be

applied to asynchronous WSNs for heterogenous energy saving requirement and meanwhile guaranteeing bounded neighbor discovery latency.

- *gqs-pair* [46]. We also developed another design for heterogenous quorum system pair. This design is called the *grid quorum system pair* (or *gqs-pair*) in which each quorum system of the pair is a grid quorum system [45].

  We prove that any two grid quorum systems can form a gqs-pair. We also show that for a gqs-pair with $\sqrt{n} \times \sqrt{n}$ grid and $\sqrt{m} \times \sqrt{m}$ grid, the average discovery delay is bounded within $\frac{(n-1)(\sqrt{n}+1)}{3\sqrt{n}} < E(Delay) < \frac{(m-1)(\sqrt{m}+1)}{3\sqrt{m}}$, while the quorum ratios are $\frac{2\sqrt{n}-1}{n}$ and $\frac{2\sqrt{m}-1}{m}$, respectively.

  Comparing with cqs-pair, gqs-pair is easier to implement since any two gqs would form a gqs-pair, which can benefit practical deployment. Meanwhile, gqs-pair has better performance in terms of average neighbor discovery latency than cqs-pair.

- Asymmetric design to improve energy efficiency in clustered WSNs.

  We observe that it is not necessary to always guarantee the intersection property for cluster members since there is usually no data transmission between the members and they do not need to discover each other in the idle state. We can reason about the data transmission as a *write operation*, and the listening in the idle state as a *read operation*. Based on this notion, we propose the concepts of read quorum and write quorum. In order to save energy, it is only necessary to guarantee the intersection property between read and write quorums, i.e., $write \cap write \neq \emptyset$, $read \cap write \neq \emptyset$ and read quorums will not necessarily intersect with each other. Hence, if a node adopts a read quorum in the idle state, and switch to a write quorum in case of data transmission, we can guarantee network connectivity, while providing higher energy efficiency.

  The asymmetric quorum-based wakeup scheduling is based on such an observation. An example design is the grid quorum group, i.e., a read quorum consists of a column of elements in a grid and a write quorum consists of a row of elements in the grid. We design a protocol, p-Grid, based on quorum groups to achieve better energy saving ratio and discovery latency, and which can be easily implemented.

  Comparing with conventional neighbor discovery protocols (such as B-MAC [13]), p-Grid is more flexible to meet heterogenous energy saving requirements and is more energy efficient over unreliable environment by avoiding continuously sending out probing messages.

- Time-dependent all-to-one shortest path routing.

  With quorum-based wakeup scheduling at the MAC layer, both the one-hop transmission latency and the E2E transmission latency are varying at different departure times. Here, we refer to the transmission latency mainly as the neighbor discovery latency after introducing the quorum-based mechanism, and not including the queuing delay and the retransmission latency. If we denote the latency of one node discovering another node as the *link cost*, then link costs are time-dependent.

In WSNs with dynamic link costs, the problem of determining an optimal routing path with the shortest latency becomes more complex after considering the time domain. The classical Bellman-Ford algorithm [47] or Dijkstra algorithm [48] cannot be used directly for finding such shortest paths because they do not consider the time domain. Thus, the built-up routing path in the last time slot will not be valid in the current time slot anymore.

We model asynchronous duty-cycled WSNs as time-dependent networks, and model the routing problem, formally, as the time-dependent Bellman-Ford (TD-Bellman) problem. We show that such networks satisfy the FIFO condition [49] and the triangular path condition. We also present distributed algorithms for finding the time-expanded shortest paths to the sink node for all nodes. When compared to the previous solution in [50], our algorithms find the shortest paths in a single execution for infinite time intervals. Additionally, we present distributed shortest path maintenance algorithms with low message complexity and space efficiency.

Comparing with past works [50], the algorithms proposed by us have lower time and message complexities in the worst case. And the number of execution times for enumerating shortest paths for all discrete time moments is bounded.

- Efficient multihop broadcast over adaptively duty-cycled WSNs.

  We consider the problem of multihop broadcast over adaptively duty-cycled WSNs where neighborhood nodes are not simultaneously awake. We present Hybrid-cast, an asynchronous and multihop broadcast protocol, which can be applied to low duty-cycling or quorum-based duty-cycling schedules, where nodes send out a beacon message at the beginning of wakeup slots. It adopts opportunistic data delivery in order to reduce the broadcast latency. We establish the upper bound of broadcast count and the broadcast latency for a given duty-cycling schedule. We evaluate Hybrid-cast through extensive simulations. The results validate the effectiveness and efficiency of our design.

  Comparing with previous solution (i.e., pure opportunistic flooding [51], and unicast replacement approach [40]), Hybrid-cast achieves better tradeoff between broadcast latency and broadcast count compared to previous broadcast solutions. Meanwhile, it reduces redundant transmission via delivery deferring and online forwarder selection.

- Prototype implementation.

  We implement a prototype system using the Telosb [10] and TinyOS 2.0 platform in order to verify the feasibility and efficiency of cqs-pair and gqs-pair in real network systems. We build the system, which contains monitoring software on a PC which acts as a base station, USB-port serial communication in the sink node side, and protocol stacks which are integrated with existing real network protocols in TinyOS for common sensors. We develop a variety of APIs using which an application initiates the parameter configuration for cas-pair or gqs-pair. Our implementation does not need to modify the upper layer routing protocols.

  We also compare the performance of our protocols with that of CSMA protocol in terms of energy-saving ratio and neighbor discovery latency in the prototype system, to show the

design tradeoff for cqs-pair and gqs-pair.

The prototype implementation reveals that our algorithms for quorum-based wakeup scheduling can be applied to real WSN platforms with desired performance, and that the implementation is easily to work with existing protocol stack and software modules.

1.4 Organization

The rest of the dissertation is organized as follows: We first summarize the past and related works in Chapter 2. In Chapter 3, we outline the basic preliminaries of quorum-based power-saving protocols. The detailed algorithms and protocol designs for asynchronous wakeup scheduling, including cqs-pair, gqs-pair, and p-Grid are discussed in Chapter 4. We present our distributed algorithms for time-varying shortest path routing in Chapter 5. The Hybrid-cast protocol for efficient multihop broadcast is described in Chapter 6. The details of our prototype implementation are discussed in Chapter 7. Finally, we conclude the dissertation and identify future research directions in Chapter 8.

Chapter 2

Past and Related Works

Since the dissertation is mainly focusing on duty-cycled WSNs, we mainly review the related works on low duty-cycling modes, neighbor discovery mechanisms under such modes, the routing mechanisms and broadcasting mechanism for low duty-cycled WSNs. We do not provide a thorough overview on research works for always-on WSNs, but a general overview for such networks can be found in [2].

2.1 Low Duty-Cycling MAC Protocols

2.1.1 LPL/ALPL mode

The LPL mode means that a node only wakes up and listens the channel state for a short time period. The example includes B-MAC [13], X-MAC [27] and S-MAC [23]. B-MAC is a CSMA-based technique that utilizes low power listening and an extended preamble to achieve low power communication. In B-MAC, nodes have an awake and a sleep period, and each node can have an independent schedule. If a node wishes to transmit, it precedes the data packet with a preamble that is slightly longer than the sleep period of the receiver. During the awake period, a node samples the medium and if a preamble is detected, it remains awake to receive the data. With the extended preamble, a sender is assured that at some point during the preamble, the receiver will wake up, detect the preamble, and remain awake in order to receive the data. The designers of B-MAC claimed that B-MAC surpasses existing protocols in terms of throughput, latency, and for most cases, energy consumption. While B-MAC performs quite well, it suffers from the overhearing problem, and the long preamble dominates the energy usage.

To overcome some of B-MAC's disadvantages, X-MAC [27] and DPS-MAC [52] have been proposed. In X-MAC or DPS-MAC, short preamble was proposed to replace the long preamble in B-MAC. Also, there is receiver information embedded in the short preamble to avoid the over-

hearing problem. The main disadvantage of B-MAC, X-MAC and DPS-MAC is the difficulty of reconfiguring the protocols after deployment, and thus lacking in flexibility. B-MAC [13], X-MAC [27] DPS-MAC [52] are compatible with LPL mechanisms. However, they do not explicitly support adaptive duty cycling where nodes choose duty cycle depending on their remained energy.

Jurdak *et. al.* [18] and Vigorito *et. al.* [15] present adaptive low power listening (ALPL) mode based on the residual energy of nodes. These works provide the application spaces for the work on time-varying routing in the dissertation. In ALPL, since nodes have heterogenous duty cycle setting, it is more difficult for neighbor discovery since a node cannot differentiate whether a neighbor is sleeping or failed when there is feed-back from the neighbor. ALPL also brings time-dependent link-cost and end-to-end distance as illustrated in Section 5.3.

2.1.2 Slotted Listening Mode

Besides LPL/ALPL mode, another duty-cycling mode is slotted listening mode as adopted in [39, 40]. Slotted listening means that a node wakes up one or more slot(s) for every n_i (n_i is an integer) time slots. In such mode, two schedules of different nodes do not always overlap on wakeup slots. In slotted listening mode, a beacon message is usually sent out in the beginning of wakeup slots, so that a sender (by staying awake for enough long time) can probe the presence of a receiver in case of data transmission.

Slotted listening mode is also adopted by works for asynchronous wakeup scheduling, such as in [11, 12, 30]. Our works [46] for quourm-based wakeup scheduling is assuming this mode.

2.2 Neighbor Discovery Mechanisms over Duty-Cycled WSNs

2.2.1 On-Demand neighbor discovery

The implementation of on-demand wakeup schemes [19, 22, 53] typically requires two different channels: a data channel and a wakeup channel for awaking nodes as and when needed. It is assumed that the nodes can be signaled and awakened at any point of time and then a message is sent to the node. This is usually implemented by employing two wireless interfaces. The first radio is used for data communication and is triggered by the second ultra low-power (or possibly passive) radio which is used only for paging and signaling. This allows for the immediate transmission of a signal on the wakeup channel if a packet transmission is in progress on the other channel, thus reducing the wakeup latency.

STEM [19] and its variation [20], and passive radio-triggered solutions [21] are examples of this class of wakeup methods. The drawback is the additional cost for the second radio. The STEM (Sparse Topology and Energy Management) work [19] uses two different radios for wakeup signals and data packet transmissions, respectively. The key idea is that a node remains awake until it has

not received any message destined for it for a certain period of time. STEM uses separate control and data channels, and hence the contention among control and data messages is alleviated. The energy efficiency of STEM is dependent on that of the control channel.

Thus, although these methods can be optimal in terms of both delay and energy, they are not yet practical. The cost issues, currently limited available hardware options which results in limited range and poor reliability, and stringent system requirements prohibit the widespread use and design of such wakeup techniques.

2.2.2 Synchronized neighbor discovery

In this class [23, 24, 28, 54–56], the nodes follow deterministic (or possibly random) wakeup patterns. Time synchronization among the nodes in the network is generally assumed. These schemes require that all neighboring nodes wake up at the same time.

The simplest way is by using a Fully synchronized pattern, like that in the S-MAC protocol [23]. In this case, all nodes in the network wakeup at the same time according to a periodic pattern. S-MAC follows a virtual clustering approach to synchronize the nodes to a common wakeup scheme with a slotted structure. By regularly broadcasting SYNC packets at the beginning of a slot, neighboring nodes can adjust their clocks to the latest SYNC packet in order to correct relative clock drifts.

In a bootstrapping phase, nodes listen for incoming SYNC packets in order to join the WSNs, and join a virtual synchronization cluster. When hearing no SYNC's, a node starts alternating in its wake-up pattern and propagates its schedule with SYNC messages. A problem of the virtual clustering arises when several clusters evolve. Bordering nodes in-between two clusters have to adopt the wake-patterns of both clusters, which imposes twice the duty cycles to these nodes. An S-MAC slot consists in a listen interval and a sleep interval. The listen interval is fragmented into a synchronization window to exchange SYNC messages, and a second and third window dedicated to RTS-CTS exchange. Nodes with receiving a RTS traffic announcement will clear the channel with a CTS respective window, and stay awake during the sleep phase, whereas all other nodes will go back to sleep.

The slot length and duty cycle must be set in a fixed manner, which severely restrains latency and maximal throughput. This can be disadvantageous, as traffic can often be of bursty nature and the rate of traffic can vary over time.

A further improvement can be achieved by allowing nodes to switch off their radio when no activity is detected for at least a timeout value, like that in the T-MAC protocol [24]. In T-MAC, the listen interval ends when no activation event has occurred for a given time threshold. An activation event may be the sensing of any communication on the radio, the end-to-end transmission of a node's data transmission, overhearing a neighbor's RTS or CTS which may announce traffic destined to itself. One drawback of T-MAC's adaptive time-out policy is that nodes often go to sleep too early.

The disadvantages of scheduled neighbor discovery schemes include the complexity and the over-

head for synchronization.

2.2.3 Asynchronous neighbor discovery

We mainly review quorum-based asynchronous neighbor discovery mechanism in this section.

quorum design. The concept of quorum systems, which are widely used in the design of distributed systems [57–62] for the application of data replicas, mutual exclusion and fault tolerance. A quorum system is a collection of sets such that the intersection of any two sets is always non-empty. There are two widely used quorum systems [45]: cyclic quorum system and grid quorum systems.

quorum-based wakeup scheduling [12, 63]. This was first introduced in [11] in the context of IEEE 802.11 ad hoc networks. The authors proposed three different asynchronous sleep/wakeup schemes that require some modifications to the basic IEEE 802.11 Power Saving Mode. More recently, Zheng *et al.* [12] took a systematic approach toward designing asynchronous wakeup mechanisms for ad hoc networks (which is also applicable for WSNs). They formulate the problem of generating wakeup schedules as a block design problem and derive theoretical bounds under different communication models. The basic idea is that each node is associated with a Wakeup Schedule Function (WSF) that is used to generate a wakeup schedule. For two neighboring nodes to communicate, their wakeup schedules must overlap regardless of their clock difference.

For the quorum-based asynchronous wakeup design, Luk and Wong [45] designed a cyclic quorum system using difference sets. However, they perform an exhaustive search to obtain a solution for each cycle length N, which is impractical when N is large.

Asymmetric quorum design [64]. In the clustered environment of sensor networks, it is not always necessary to guarantee all-pair neighbor discovery. The Asymmetric Cyclic Quorum (ACQ) system [64] was proposed to guarantee neighbor discovery between each member node and the clusterhead, and between clusterheads in a network. The authors also presented a construction scheme which assembles the ACQ system in $O(1)$ time to avoid exhaustive searching. ACQ is a generalization of the cyclic quorum system. The scheme is configurable for different networks to achieve different distribution of energy consumption between member nodes and the clusterhead.

However, it remains a challenging issue to efficiently design an asymmetric quorum system given an arbitrary value of n. One previous study [12] shows that the problem of finding an optimal asymmetric block design can be reduced to the minimum vertex cover problem, which is NP-complete. However, the ACQ [64] construction is not optimal in that the quorum ratio for symmetric-quorum is $\phi = \lceil \frac{n+1}{2} \rceil$ and the quorum ratio for asymmetric-quorum is $\phi' = \lceil \sqrt{\frac{n+1}{2}} \rceil$. Another drawback is that it cannot be a solution to the h-QPS problem since the two asymmetric-quorums cannot guarantee the intersection property.

Transport layer approach. Wang *et al.* [65] applied quorum-based wakeup scheduling at the transport layer which can cooperate with any MAC-layer protocol, allowing for the reuse of well-understood MAC protocols. The proposed technique saves idle energy by relaxing the requirement

for end-to-end connectivity during data transmission and allowing the network to be disconnected intermittently via scheduled sleeping. The limitation of this work is that each node adopts the same grid quorum system as its wakeup schedule, and the quorum ratio is not optimal when compared with that of cyclic quorum systems.

Schedules based on Chinese Remainder Theorem. In [63], the authors present a mechanism called Disco which is a simple adaptation of the Chinese Remainder Theorem [66]. They show that Disco can ensure asynchronous neighbor discovery in bounded time, even if nodes independently set their own duty cycles. Another work [67] called C-MAC adopts similar mechanism for wakeup scheduling in WSNs.

In [68], Kuo *et. al.* adopt relative primes as the wakeup schedules for neighbor nodes in order to support multicast in asynchronous wakeup mechanisms. The main principle is the intersection property from Chinese Remainder Theorem [66]. The limitation of this mechanism is that the discovery latency is usually too long, i.e., over 100 slots for a $(13, 17)$ prime pair in [63], to satisfy the delay constraints of many WSN applications, which prevent their practical applications.

2.3 Routing over Duty-cycled WSNs

2.3.1 Opportunistic Routing

Traditional routing protocols select the optimal path for each destination and forward a packet to the corresponding next hop. While such optimal-path routing schemes are considered well-suited for networks with reliable point-to-point links, they are not necessarily ideal for wireless networks with lossy broadcast links. Consequently, opportunistic routing schemes [69] that exploit the broadcast nature of wireless transmissions and dynamically select a next-hop per-packet based on loss conditions at that instant are being actively explored.

Different opportunistic routing protocols have been proposed recently for routing in WSNs. These protocols exploit the redundancy among nodes by using a node that is available for routing at the time of packet transmission. Biswas *et. al.* [69] proposed EXOR which utilizes the broadcast nature of wireless medium by selecting the next forwarder among those which successfully received data after data transmission.

The opportunistic routing mechanism was theoretically studied in [70] by analytically modeling the delay performance given random duty cycle setting. In [71], the authors analyze the efficacy of opportunistic routing, and define a new metric EAX (expected any-path transmissions) that captures the expected number of any-path transmissions needed to successfully deliver a packet between two nodes under opportunistic routing. Based on EAX, the authors propose a candidate selection and prioritization method corresponding to an ideal opportunistic routing scheme.

Ye *et. al.* [39] proposed data forwarding mechanism with opportunistic loops to improve transmission reliability in WSNs. Instead of ETX (Expected Transmission Count) metric, the authors

design a new data delivery method to optimize source-to-sink data delivery ratio, E2E delay, or energy consumption under unreliable and intermittent connectivity within scheduled networks. The work combines effect of sleep latency and unreliable communication links, which dramatically reduces the effectiveness of the existing solutions. A novel dynamic switch-based forwarding technique over time dependent networks is proposed to achieve optimal expected delivery ratio (EDR), expected E2E delay (EED), or expected energy consumption (EEC), respectively.

Opportunistic routing mitigates the effect of varying channel conditions and duty cycling that make static selection of routes not viable. However, as pointed out in [72], there is a downside as each hop may provide extremely small progress towards the destination or the signaling overhead for selecting the forwarding node may be too large.

2.3.2 Deterministic Routing

As to deterministic routing, it includes shortest path routing, minimum-hop routing, on-demand routing (AODV), geographic routing etc,. However, we only review the deterministic time-dependent shortest path algorithms and the maintenance algorithms, because they are related tightly with our works in the dissertation.

Time-Dependent Shortest Path Problem: This problem was first proposed by Cooke and Halsey [73]. It has been well studied in the field of traffic networks [49], time-dependent graphs [74], and GPS navigation [75]. Previous solutions for time-dependent shortest path problem mostly works offline by a centralized approach [74]. Although these solutions can provide inspirations, they cannot be applied to WSNs where global network topology is not known by a centralized node given the large-scale size of a WSN.

For the distributed time-dependent shortest path problem, the only previous work [50] computes the shortest paths for a specific departure time in each execution, which is not time-efficient. If the whole time period has M discrete intervals (M is ∞ for infinite time intervals), we have to execute the algorithm in [50] M times, which is inefficient in terms of message complexity and time complexity. After multiple execution, the algorithm in [50] bring high message cost, which is undesirable for resource-limited WSNs.

The previous works discussed two policies [50] for time-dependent shortest path problems: waiting and non-waiting. The waiting do not means waiting in the buffer, but means waiting after the data delivery is available (i.e. the receiver is awake). We do not consider waiting policy in our works since the end-to-distance will not benefit from the waiting.

Dynamic Shortest-Path maintenance: Many works [76–78] exist for handling link decreases and increases, and node deletions and insertions in static networks. In [79], an algorithm is given for computing all-pairs shortest paths requiring $O(n^2)$ messages when the network size is n. In [80], an efficient incremental solution has been proposed for the distributed all-pairs shortest paths problem, requiring $O(nlog(nW))$ amortized number of messages over a sequence of edge insertions and edge weight decreases. Here, W is the largest positive integer edge weight. In [81], Awerbuch *et*

al. propose a general technique that allows to update the all-pairs shortest paths in a distributed network in $O(n)$ amortized number of messages and $O(n)$ time, by using $O(n^2)$ space per node.

In [78], Ramarao and Venkatesan give a solution for updating all-pairs shortest paths that requires $O(n^3)$ messages and time and $O(n)$ space. They also show that, in the worst case, the problem of updating shortest paths is as difficult as computing shortest paths. The results in [78] have a remarkable consequence. They suggest that two possible directions can be investigated in order to devise efficient fully dynamic algorithms for updating all-pairs shortest paths: 1. to study the trade-off between the message, time and space complexity for each kind of dynamic change; 2. to devise algorithms that are efficient in different complexity models (with respect to worst case and amortized analysis).

However, these algorithms [77, 78] need $O(n)$ space at each node, which is impractical for sensor nodes with limited memory capacity. In addition, none of the previous works are efficient for shortest path maintenance in time-dependent networks.

2.4 Broadcast over Duty-Cycled WSNs

2.4.1 Gossip or Opportunistic Approach

Opportunistic unicast routing, like EXOR [69], was proposed to exploit wireless broadcast medium and multiple opportunistic paths for efficient message delivery. Regarding broadcasting, the main purpose of opportunistic approach aimed at ameliorating message implosion. Smart Gossip [82] adaptively determines the forwarding probability for received flooding messages at individual sensor nodes based on previous knowledge and network topology.

In Opportunistic Flooding [51] (abbreviated as OppFlooding), each node makes probabilistic forwarding decisions based on the delay distribution of next-hop nodes. Only opportunistic early packets are forwarded via the links outside of the energy-optimal tree to reduce flooding delays and the level of redundancy. To resolve decision conflicts, the authors build a reduced flooding sender set to alleviate the hidden terminal problem. Within the same sender set, the solution uses a link-quality-based backoff method to resolve and prioritize simultaneous forwarding operations. The main problem of pure opportunistic flooding is the overhead in terms of transmission times.

2.4.2 Synchronized or Duty-cycle Awareness

Wang et al. [83] present a centralized algorithm, mathematically modeling the multihop broadcast problem as a shortest-path problem in a time-coverage graph, and also present two similar distributed algorithms. However, their work simplifies many aspects necessary for a complete MAC protocol, and may not be appropriate for real implementation. The work also assumes duty-cycle awareness, which makes it difficult to use it in asynchronous WSNs since duty-cycle awareness

needs periodic time-synchronization due to clock drifting. RBS [84] proposes a broadcast service for duty-cycled sensor networks and shows its effectiveness in reducing broadcast count and energy costs.

All these works based on synchronization assume that there are usually multiple neighbors available at the same time to receive the multicast/flooding message sent by a sender. This is not true in low duty-cycled asynchronous networks.

2.4.3 Asynchronous Mechanisms

B-MAC [13] can support single-hop broadcast in the same way as it supports unicast, since the preamble transmission over an entire sleep period gives all of the transmitting node's neighbors a chance to detect the preamble and remain awake for the data packet. X-MAC [27] substantially improves B-MAC's performance for unicast, but broadcast support is not clearly discussed in that work. X-MAC is not promising for broadcast since the transmitter has to continually trigger the neighbors to wake up.

ADB [40] avoids the problems faced by B-MAC and X-MAC by efficiently delivering information on the progress of each broadcast. It allows a node to go to sleep immediately when no more neighbors need to be reached. ADB is designed to be integrated with an unicast MAC that does not occupy the medium for a long time, in order to minimize latency before forwarding a broadcast. The effort in delivering a broadcast packet to a neighbor is adjusted based on link quality, rather than transmitting throughout a duty cycle or waiting throughout a duty cycle for neighbors to wake up. Basically, ADB belongs to the unicast replacement approach and it needs significant modification to existing MAC protocols for supporting broadcast.

Chapter 3

Preliminaries, Assumptions and Problem Statement

3.1 Network Model and Assumptions

We represent a multi-hop wireless sensor network by a directed graph $G(V, E)$, where V is the set of network nodes ($|V|$ = network size), and E is the set of edges. If node v_j is within the transmission range of node v_i, then an edge (v_i, v_j) is in E. We assume bidirectional links. We define the one-hop neighborhood of node n_i as $N(i)$.

In the dissertation, we use the term "connectivity" loosely, in the sense that a topologically connected network in our context may not be connected at any time; instead, all the nodes are reachable from a node within a finite amount of time.

We assume two low duty-cycling mode, LPL/ALPL and slotted listening. For both modes, time axes are arranged as consecutive short time slots, all slots have the same duration in a node. In LPL/ALPL mode, at the beginning of a time interval, a node will check the state of its channel. In slotted listening mode, all slots have the same duration T_s, and each node n_i adopts a periodic wakeup schedule every L_i time slots. The wakeup schedule can be once every L_i slots or based on quorum schedules (i.e., cyclic quorum systems or grid quorum systems [45]). L_i is called cycle length for node n_i. We assume that beacon messages are sent out at the beginning of wakeup slots in slotted listening mode, as assumed in [1, 85]. When a node wants to transmit data, it will wait until beacons are received from receivers.

We also make the following assumptions: (1) There is no time synchronization between nodes (thus the time slots in two nodes are not necessarily aligned); (2) The overhead of turning on and shutting down radio is negligibly small compared with the long duration of time slots (i.e., $50ms \sim 500ms$); (3) There is only one sink node in the network (but our works can be easily extend to the scenario of multiple sink nodes).

In all our following works, we do not explicitly consider radio interference in the problem modeling. We consume the interference can be concealed by sleeping, RTS/CTS mechanism, and multiple radio or frequencies.

As to the length of one time interval, it depends on application-specific requirements or energy-saving requirement. For example, for a radio compliant with IEEE 802.15.4 [86,87], the bandwidth is approximately 128kb/s and a typical packet size is less than 512KB. Given this, the slot length (i.e., the beacon interval) can be approximately $50ms$.

3.2 Quorum-based wakeup scheduling

3.2.1 Quorum Systems

We use the following definitions for quorum systems. Given a cycle length n, let $U = \{0, \cdots, n-1\}$ be an universal set.

Definition 1 *A quorum system $\mathcal{Q}$ under U is a superset of non-empty subsets of U, each called a quorum, which satisfies the intersection property: $\forall G, H \in \mathcal{Q} : G \cap H \neq \emptyset$.*

Definition 2 *Given an integer $i \geq 0$ and quorum G in a quorum system $\mathcal{Q}$ under U, we define $G + i = \{(x + i) \; mod \; n : x \in G\}$.*

Definition 3 *A quorum system $\mathcal{Q}$ under U is said to have the* rotation closure property *if $\forall G, H \in \mathcal{Q}$, $i \in \{0, 1, ...n-1\}$: $G \cap (H + i) \neq \emptyset$.*

There are two widely used quorum systems, *grid quorum system* and *cyclic quorum system*, that satisfy the *rotation closure property*.

Grid quorum system [45]. In a grid quorum system, shown in Figure 3.1, elements are arranged as a $\sqrt{n} \times \sqrt{n}$ array (square). A quorum can be any set containing a column and a row of elements in the array. The quorum size in a square grid quorum system is $2\sqrt{n} - 1$. An alternative is a "triangle" grid-based quorum in which all elements are organized in a triangle fashion. The quorum size in "triangle" quorum system is approximately $\sqrt{2}\sqrt{n}$.

Cyclic quorum system [45]. A cyclic quorum system is based on the ideas of cyclic block design and cyclic difference sets in combinatorial theory [44]. The solution set can be strictly symmetric for arbitrary n. For example, the set $\{1, 2, 4\}$ is a quorum from a cyclic quorum system with cycle length $= 7$. Figure 3.1 illustrates three quorums from a cyclic quorum system with cycle length 7.

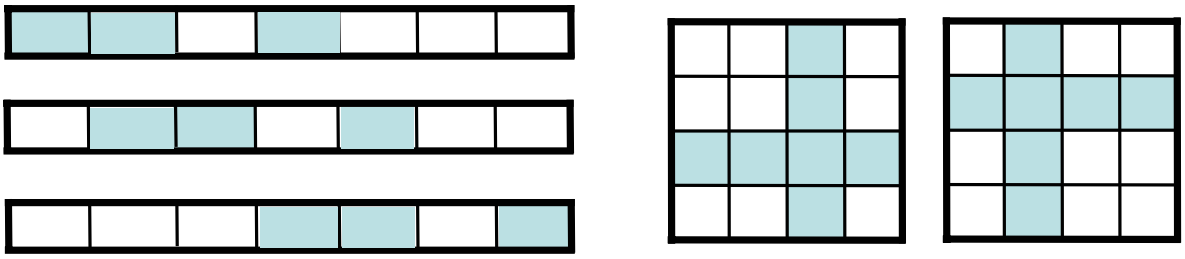


Figure 3.1: Cyclic Quorum System (left) and Grid Quorum System (right)

3.2.2 Quorum-based Power Saving

Previous work [30] has defined the **QPS** (quorum-based power-saving) problem as follows: Given an universal set $U = \{0, 1, ...n-1\}$ ($n > 2$) and a quorum system $\mathcal{Q}$ over U, two nodes that select any quorum from $\mathcal{Q}$ as their wakeup schedules must have at least one overlap in every n consecutive time slots.

Theorem 1 *$\mathcal{Q}$ is a solution to the QPS problem if $\mathcal{Q}$ is a quorum system satisfying the* rotation closure property.

Theorem 2 *Both grid quorum systems and cyclic quorum systems satisfy the rotation closure property and can be applied as a solution for the QPS problem in WSNs.*

Proofs of Theorems 1 and 2 can be found in [30].

Since sensor nodes are subject to clock drift, the time slots are not exactly aligned to their boundaries in practical deployments. In most cases, two nodes only have *partial overlap* during a certain time interval. It has been shown that two nodes that adopt quorum-based wakeup schedules can discover each other even under *partial overlap*.

Theorem 3 *[12] If two quorums ensure a minimum of one overlapping slot, then with the help of a beacon message at the beginning of each slot, two neighboring nodes can hear each others' beacons at least once.*

Theorem 3's proof is presented in [12]. An illustration is given in Figure 3.2. Suppose that node A's quorum and node B's quorum intersect with each other in the first element and that the clock drift between the two nodes is Δt ($1\ slot < \Delta t < 2\ slots$). We can see that node A's 1^{st} beacon message in the current cycle (beacon messages are marked with solid lines) will be heard by node B during node B's 2^{nd} time slot interval in its current cycle. Meanwhile, node B's 2^{nd} beacon message in the current cycle will be heard by node A during its n^{th} time slot interval in the previous cycle (beacon messages are marked with dash lines).

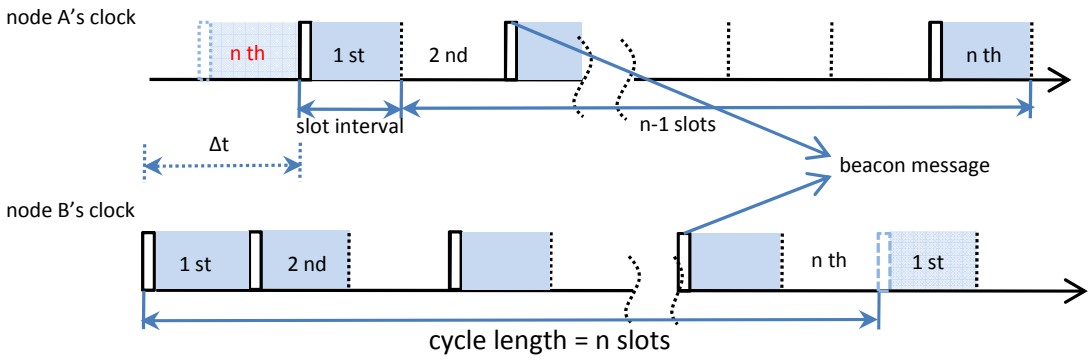


Figure 3.2: Neighbor discovery under partial overlap

This theorem ensures that two neighboring nodes can always discover each other within bounded time if all beacon messages are transmitted successfully. This property also holds true even in the case when two originally disconnected subsets of nodes join together as long as they use the same quorum system.

3.3 Neighbor Discovery Mechanism with LPL mode

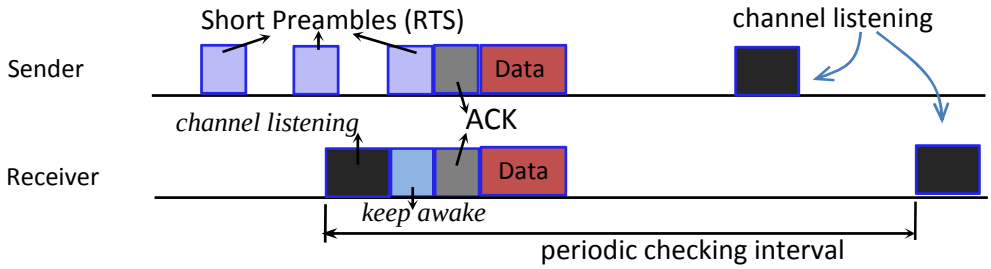


Figure 3.3: Neighbor discovery mechanism with periodic LPL scheduling in the X-MAC protocol

Previous works on low power listening (or LPL) adopt periodic preamble sampling mechanisms [13, 27] in which a node checks the state of its channel once every x time units, where x is usually $100ms$ or $200ms$. If the gain of the channel is less than a certain threshold level, it means that there is no activity from its neighbors and the node will go back to sleep.

When a sender wants to send out data, it first sends out a long preamble [13] or multiple short strobed preambles, which contain the sender's identity [27]. When the desired receiver detects the short preamble, it will keep awake and will feed back an acknowledgment to the sender. After the sender receives the acknowledgement, the actual data transmission will begin.

An illustration is given in Figure 3.3. In the idle state, both the sender and the receiver follow

periodic LPL scheduling. Once the sender wants to transmit data, it sends out multiple short strobed preambles to trigger the receiver to wake up.

3.4 Chinese Remainder Theorem

Theorem 4 *[88]. Let p_1 , p_2 , $\cdots$, p_m be m positive integers which are pairwise relatively prime, i.e., $gcd(p_i, p_j) = 1$ (greatest common divisor of p_i and p_j) when $i \neq j$. Let $P = \prod_{i=1}^{m} p_i$ and let r_1, $\cdots$, r_m be m integers. Then the system of linear congruences*

$$I \equiv r_1 (mod\ p_1) \equiv r_2 (mod\ p_2) ... \equiv r_m (mod\ p_m)$$

has a common solution to all the congruences, and any two solutions are congruent to one another modulo P. Furthermore, there exists exactly one solution I between 0 and P.

For example, consider $r_1 = 1$ and $r_2 = 3$, and $p_1 = 3$ and $p_2 = 5$. We can have $x = 13 < 3 \times 5$, which satisfy the following congruences: $x \equiv\ 1\ (mod\ 3)$ and $x \equiv\ 3\ (mod\ 5)$.

We can apply the Chinese Remainder Theorem for wakeup scheduling in WSNs: two neighbor nodes can choose pairwise relatively primes as their wakeup schedules, i.e., one node wakes-up once every 3 consecutive time slots and another node wakes-up once every 5 consecutive time slots. Then, we can guarantee that they must have overlapped awake slots within 3×5 consecutive slots regardless of the clock drift.

3.5 Problem Statement and Goals

The main goal of the works in the dissertation contains three aspects: 1. To maintain network connectivity in duty-cycled network; and 2. To design a fast distributed algorithm for the time-varying shortest path routing and path maintenance; and 3. To broadcast message asynchronously to entire network with low flooding latency and low message cost.

Specifically, we introduce the following problems to be solved in the dissertation.

3.5.1 Heterogeneous Quorum-Based Wakeup Scheduling

We introduce the h-QPS (heterogeneous quorum-based power saving) problem in this section [1]. In WSNs, it is often desirable that different nodes wakeup according to heterogeneous quorum-based schedules. There are several reasons for this. First, many WSNs have heterogeneous nodes such as cluster-heads, gateways, and relay nodes [89]. They often have different requirements regarding average neighbor discovery delay and energy saving ratio. For cyclic quorum systems,

generally, cluster-heads should wakeup based on a quorum system with small cycle length, and member nodes should wakeup based on a longer cycle length. Second, WSNs that are used in applications such as environment monitoring typically have seasonally-varying power saving requirements. For example, a sensor network for wild fire monitoring may require a larger energy saving ratio during winter seasons. Thus, they often desire variable cycle-length wakeups during different seasons.

We define the **h-QPS** problem as follows. Given two heterogeneous quorum systems $\mathcal{X}$ over $\{0, 1, \cdots, n-1\}$ and $\mathcal{Y}$ over $\{0, 1, \cdots, m-1\}$ ($n \leq m$), design a pair $(\mathcal{X}, \mathcal{Y})$ in order to guarantee that:

1. two nodes that select two quorums $G \in \mathcal{X}$ and $H \in \mathcal{Y}$ as their wakeup schedules, respectively, can hear each other at least once within every m consecutive slots; and
2. $\mathcal{X}$ and $\mathcal{Y}$ are solutions to QPS, individually.

A solution to the h-QPS problem is important toward ensuring connectivity when we want to dynamically change the quorum systems between all nodes. For example, suppose that all nodes in a WSN initially wakeup via a larger cycle length. When there is a need to reduce the cycle length (e.g., to meet a delay requirement or due to changing seasons), the sink node can send a request to the whole network gradually through some relay nodes. The connectivity between a relay node and the remaining nodes will be lost if the relay node first changes its wakeup schedule to a new quorum schedule, which cannot overlap with the original schedules of the remaining nodes.

Although grid quorum systems and cyclic quorum systems can be applied as a solution for the QPS problem, that does not necessarily mean that any pair of such systems can be a solution to the h-QPS problem. We will show this in Section 4.1.2.

3.5.2 Time-Dependent Bellman-Ford Equation

We model a WSN as a directed graph $G = (V, E, C)$, with $|V|$ nodes and $|E|$ links. $C = \{\tau_{i,j}(t) | (i, j) \in E\}$ is a set of time-dependent link delays, i.e., $\tau_{i,j}(t)$ is a strictly positive function of time defined for $[0, \infty)$, describing the delay of a message over link (i, j) at time t. Each node n_i only knows the identity of the nodes in its neighbor set, defined as N_i.

We assume that time axes is arranged as consecutive short time slots. We denote the duration of one time slot for node n_i as T_i. It is possible that $T_i \neq T_j$ (ALPL) for two nodes n_i and n_j. The time expansion of each node n_i is modeled as discrete and infinite, where $\mathfrak{T}_i = \{t_i^0, t_i^1, t_i^2, \cdots, t_i^M\}$, M is $+\infty$, and $t_i^k - t_i^{k-1} = T_i$. We use the terms of checking interval and time slot interchangeably.

The wakeup schedule depends the underlying MAC protocols. We first assume a node can be operated with LPL mode where a node wakes up at the beginning of a time slot to check the channel state. If there is no activity, the node goes back to sleep, otherwise, it should stay awake. Then we relax the assumption and discuss how our works can be applied to other wakeup schedules like quorum schedules [1].

We consider the policy of no waiting at each node since the node-to-sink delay will not benefit from waiting. Thus, once the data arrives at an intermediate node, the node will try to dispatch the data immediately. Dispatching times are not the same as the data departure times, as the data may still be buffered in the sender's memory. For simplicity in modeling and design, a node dispatches the received data at $t_i^k \in \mathfrak{T}_i$.

A nonnegative travel time $\tau_{i,j}(t_i^k)$ is associated with each link (i, j) with the following meaning: If t_i^k is the data dispatching time from node n_i along the link (i, j), then $t_i^k + \tau_{i,j}(t_i^k)$ is the data arrival time at node n_j.

The general problem of determining the shortest paths with the least latency in time-dependent WSNs can be defined as follows: Find the least time paths from all nodes to the sink node n_s corresponding to the minimum achievable delay d_i, $\forall n_i \in V$ and $\forall t_i^k \in \mathfrak{T}_i$, where

$$d_i(t_i^k) = \min_{n_j \in N_i} \{\tau_{i,j}(t_i^k) + d_j[t_i^k + \tau_{i,j}(t_i^k)]\} \tag{3.1}$$

Equation 5.1 is an extension of Bellman's equations [90] for the time-dependent network and is referred to as TD-Bellman's equation hereafter.

3.5.3 Hybrid Broadcast

Let us define the broadcast latency as the time between the beginning of a broadcast and the time at which every node receives the broadcast message. Also, let us define the broadcast count as the number of broadcasting via all nodes to ensure that the entire network receives the message.

The goal for supporting efficient multi-hop broadcasting is to design a broadcast schedule, which can not only shorten the broadcast latency but also reduce the broadcast count for flooding a message to the entire network. The protocol that we will present, Hybrid-cast, is a heuristic solution to this problem.

Chapter 4

Asynchronous Wakeup Scheduling

In this chapter, we first present our work on heterogenous quorum-based wakeup scheduling, then introduce our design on asymmetric quorum-based wakeup scheduling. The major distinction between the two mechanisms is that any two different quorums have non-empty intersection for the first work, but in the second work, some quorums such as read quorums do not satisfy this property.

4.1 Heterogenous Quorum-based Wakeup Scheduling

4.1.1 Heterogeneous Rotation Closure Property

First, we define a few concepts to facilitate our presentation.

Definition 4 *(*p*-extension). Given two positive integers n and p, for a set $A = \{a_i | 1 \leq i \leq k, a_i \in \mathbb{Z}_n\}$, the* p-extension *of A is defined as $A^p = \{a_i + j * n | 1 \leq i \leq k, 0 \leq j \leq p-1, a_i \in \mathbb{Z}_n\}$. For a quorum system $\mathcal{Q} = \{A_1, \cdots, A_m\}$, the* p-extension *of $\mathcal{Q}$ is defined as $\mathcal{Q}^p = \{A_1^p, \cdots, A_m^p\}$.*

Since time axes is infinite, the physical meaning of *p*-extension of a schedule is same as the original schedule. Thus, *p*-extension is just a different logical presentation for a specified schedule of a quorum system. Example: Let $A = \{1, 2, 4\}$ in $(\mathbb{Z}_7, +)$. Now, $A^3 = \{1, 2, 4, 8, 9, 11, 15, 16, 18\}$ in $(\mathbb{Z}_{21}, +)$. If a quorum system $\mathcal{Q} = \{\{1, 2, 4\}, \{2, 3, 5\}, \cdots, \{7, 1, 3\}\}$, then we have $\mathcal{Q}^2 = \{\{1, 2, 4, 8, 9, 11\}, \{2,$
$3, 5, 9, 10, 12\}, \{3, 4, 6, 10, 11, 13\}, \cdots, \{7, 1, 3, 14, 8, 10\}\}$.

Definition 5 (Heterogeneous rotation closure property). *Given two positive integers N and M where $N \leq M$ and $p = \lceil \frac{M}{N} \rceil$, consider two quorum systems $\mathcal{X}$ over the universal set $\{0, \cdots N-1\}$*

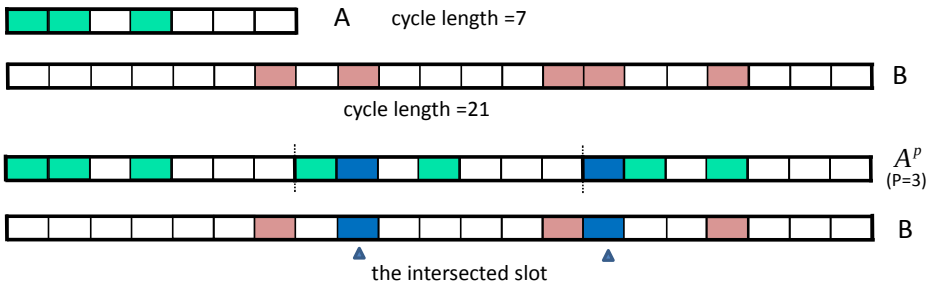


Figure 4.1: Heterogenous rotation closure property between two cyclic quorum systems: A with cycle length of 7 and B with cycle length of 21. A quorum from A's p-extension A^p will overlap with a quorum from B.

and $\mathcal{Y}$ over the universal set $\{0, \cdots M-1\}$. The pair ($\mathcal{X}$,$\mathcal{Y}$) is said to satisfy the heterogeneous rotation closure property *if :*

1. $\forall G \in \mathcal{X}^p, H \in \mathcal{Y}, i \in N+$: $G \cap (H+i) \neq \emptyset$, *and*
2. $\mathcal{X}$ *and* $\mathcal{Y}$ *satisfy the* rotation closure property *(Definition 1), respectively.*

Example: Let $A = \{1, 2, 4\}$ in $(\mathbb{Z}_7, +)$ and $B = \{1, 2, 4, 10\}$ in $(\mathbb{Z}_{13}, +)$. Consider two cyclic quorum systems $\mathcal{Q}_\mathcal{A} = C(A, \mathbb{Z}_7)$ and $\mathcal{Q}_\mathcal{B} = C(B, \mathbb{Z}_{13})$. Now, ${\mathcal{Q}_\mathcal{A}}^2 = C(\{1, 2, 4, 8, 9, 11\}, \mathbb{Z}_{14})$. We can verify that any two quorums from ${\mathcal{Q}_\mathcal{A}}^2$ and $\mathcal{Q}_\mathcal{B}$ must have non-empty intersection. Thus, the pair ($\mathcal{Q}_\mathcal{A}$,$\mathcal{Q}_\mathcal{B}$) satisfies the *heterogeneous rotation closure property*.

Lemma 1 *If two quorum systems $\mathcal{X}$ and $\mathcal{Y}$ satisfy the heterogeneous rotation closure property, then the pair $(\mathcal{X}, \mathcal{Y})$ is a solution to the h-QPS problem.*

Proof 1 *According to Definition 5, if two quorum systems $\mathcal{X}$ and $\mathcal{Y}$ satisfy the heterogeneous rotation closure property, a quorum G from $\mathcal{X}$ and a quorum H from $\mathcal{Y}$ must overlap at least once within the larger cycle length of $\mathcal{X}$ and $\mathcal{Y}$. Thus, two nodes can hear each other if they select G and H as their wakeup schedules, respectively, based on Theorem 3. This implies that $(\mathcal{X}, \mathcal{Y})$ is a solution to the h-QPS problem.*

Example: In Figure 4.1, there are two cyclic quorum systems $C(A, \mathbb{Z}_7)$ and $C(B, \mathbb{Z}_{21})$. Since they have different cycle lengths, we extend A's cycle by 3 ($3 = \lceil \frac{21}{7} \rceil$) times and denote its extension as A^p. Now, A^p will have an intersection with B within 21 time slot intervals. We can further verify that B and its rotations will overlap with A^p. Thus, $(C(A, \mathbb{Z}_7), C(B, \mathbb{Z}_{21}))$ has the *heterogeneous rotation closure property* and it can be a solution to the h-QPS problem.

Negative example: In Figure 4.2, $A = \{3, 5, 6\}$ and $B = \{7, 9, 14, 15, 18\}$ are from two cyclic quorum systems $C(A, \mathbb{Z}_7)$ and $C(B, \mathbb{Z}_{21})$. We extend A's cycle by 3 ($3 = \lceil \frac{21}{7} \rceil$) times and denote its extension as $A^p = \{3, 5, 6, 10, 12, 13, 17, 19, 20\}$. Now, $A^p \bigcap B = \emptyset$, which means that

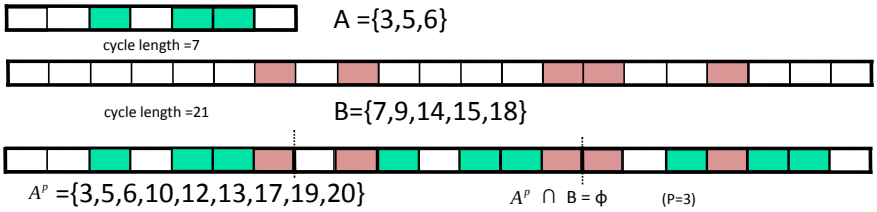


Figure 4.2: Two quorums do not satisfy heterogenous rotation closure property although they are from cyclic quorum systems respectively.

$(C(A, \mathbb{Z}_7), C(B, \mathbb{Z}_{21}))$ does not stratify the *heterogeneous rotation closure property* and it can NOT be a solution to the h-QPS problem.

4.1.2 Cyclic Quorum System Pair: cqs-pair

In this section, we present one design of heterogenous quorum systems: *cqs-pair* which is based on the cyclic block design concept and cyclic difference sets in combinatorial theory [44]. We first review two definitions which were originally introduced in [45].

Definition 6 (N, k, λ)**- *difference set.*** *A set D :* $\{a_1, ..., a_k\}(\mod N)$*,* $a_i \in [0, N-1]$*, is called a* (N, k, λ)*-difference set if for every* $d \neq 0$ *, there are* exactly λ *ordered pairs* (a_i, a_j)*,* $a_i, a_j \in D$ *such that* $a_i - a_j \equiv d\ (mod N)$*.*

We now introduce a new definition which is extended from (N, k, λ)- difference set.

Definition 7 (N, k, M, l)***-difference pair.*** *Suppose* $N \leq M$ *and* $p = \lceil \frac{M}{N} \rceil$*. Suppose there are sets* $A : \{a_1, \cdots a_k\}$ *in (*$\mathbb{Z}_N, +$*) and* $B : \{b_1, \cdots b_l\}$ *in (*$\mathbb{Z}_M, +$*). The pair* (A, B) *is defined as a* (N, k, M, l)*-difference pair if* $\forall d \in \{0, \cdots, M-1\}$*, there exists at least one ordered pair* $b_i \in B$ *and* $a_j^p \in A^p$ *such that* $b_i - a_j^p \equiv\ d\ (mod\ M)$*.*

Consider an example where $A = \{1, 2, 4\}$ and $B = \{1, 3, 6, 7\}$ be two subsets in $(\mathbb{Z}_7, +)$ and $(\mathbb{Z}_{13}, +)$, respectively. Then (A, B) is a $(7, 3, 13, 4)$-difference pair, since for A^2 ($\{1, 2, 4, 8, 9, 11\}$) and B, there exists at least one ordered pair $b_i \in B$ and $a_j^p \in A^p$ such that $b_i - a_j^p \equiv\ d\ (mod\ M)$ for $\forall d \in \{0, \cdots, M-1\}$.

$$\begin{array}{llll} 1 \equiv 3-2 & 2 \equiv 6-4 & 3 \equiv 1-11 & 4 \equiv 6-2 \\ 5 \equiv 6-1 & 6 \equiv 7-1 & 7 \equiv 3-9 & 8 \equiv 6-11 \\ 9 \equiv 7-11 & 10 \equiv 1-4 & 11 \equiv 6-8 & 12 \equiv 1-2 \\ 13 \equiv 1-1 & & & \end{array} (mod\ 13)$$

Definition 8 ***cyclic quorum system pair (cqs-pair).*** *Given two cyclic quorum $\mathcal{X} = C(A, \mathbb{Z}_N)$ and $\mathcal{Y} = C(B, \mathbb{Z}_M)$, suppose $N \leq M$. We call $(\mathcal{X}, \mathcal{Y})$ a* cqs-pair *if: $\forall (A+i)^p \subseteq \mathcal{X}^p$ and $(B+j) \subseteq \mathcal{Y}$, $(A+i)^p \cap (B+j) \neq \emptyset$.*

Theorem 5 *Given two cyclic quorum $\mathcal{X} = C(A, \mathbb{Z}_N)$ and $\mathcal{Y} = C(B, \mathbb{Z}_M)$ ($N \leq M$), the pair $(\mathcal{X}, \mathcal{Y})$ is a* cqs-pair *if and only if: (A, B) is a (N, k, M, l)-difference pair.*

Proof 2 *We first prove the following claim (sufficient condition): if (A, B) is a (N, k, M, l)-difference pair, we have $\forall (A+i)^p \subseteq \mathcal{X}^p$ and $(B+j) \subseteq \mathcal{Y}$, $(A+i)^p \cap (B+j) \neq \emptyset$. Without loss of generality, we assume that $j > i$ regarding two sets B_i and A^p_j, where $p = \lceil \frac{M}{N} \rceil$. Consider the r^{th} element of B_i and s^{th} element of A^p_j, denoted by $b_{i,r}$ and $a^p_{j,s}$, respectively. We will now show that $b_{i,r} = a^p_{j,s}$.*

Let the r^{th} element of B be b_r and the s^{th} element of A^p be a^p_s. Then $b_{i,r} - a^p_{j,s} = (b_r - a^p_s + i - j) \; mod \; M$. According to the definition of (N, k, M, l)-difference pair, there must be some r and s such that $b_r - a^p_s \equiv j - i \; (mod \; M)$. Therefore, we can always choose a pair of r and s such that $b_{i,r} - a^p_{j,s} \equiv 0 \; (mod \; M)$. This implies that $B_j \cap A^P_i \neq \emptyset$.

Now we prove the another claim (necessary condition): if $\forall (A+i)^p \subseteq \mathcal{X}^p$ and $(B+j) \subseteq \mathcal{Y}$, $(A+i)^p \cap (B+j) \neq \emptyset$, we have that$(A, B)$ is a (N, k, M, l)-difference pair. We prove the necessity by contradiction. Assume that $B_j \cap A^P_i \neq \emptyset$, but (A, B) is not a (N, k, M, l)-difference pair. Then, there exists a number $\in \{0, \cdots, M-1\}$, say t, for which $b_i - a^p_j \neq t \; (mod \; M)$, $\forall i, j$.

Consider the r^{th} element of B_t and the s^{th} element of A^p. We have $b_{t,r} - a^p_s \equiv b_r - a^p_s + t \; (mod \; M)$. Since $B_t \cap A^P_i \neq \emptyset$, $b_{t,r} - a^p_s = 0$ for some r and s. This implies that $b_r - a^p_s \equiv t \; (mod \; M)$ for some r and s, which contradicts the derivation of $b_i - a^p_j \neq t \; (mod \; M)$ $\forall i, j$ from the assumption. Therefore, the theorem holds.

Corollary 1 *Given a cyclic quorum system $\mathcal{X}$, $(\mathcal{X}, \mathcal{X})$ is a cqs-pair.*

Theorem 6 *The cyclic quorum system pair (*cqs-pair*) is a solution to the h-QPS problem.*

Proof 3 *According to the definition of cqs-pair, a cqs-pair satisfies the heterogeneous rotation closure property. Thus, the cqs-pair can be a solution to the h-QPS problem according to Lemma 1.*

Example 1: Let $A = \{1, 2, 4\}$ and $\mathcal{X} = C(A, \mathbb{Z}_7)$; $B = \{7, 9, 14, 15, 18\}$ and $\mathcal{Y} = C(B, \mathbb{Z}_{21})$. The pair $(\mathcal{X}, \mathcal{Y})$ is a cqs-pair, as illustrated in Figure 4.1. Also, both $(\mathcal{X}, \mathcal{X})$ and $(\mathcal{Y}, \mathcal{Y})$ are cqs-pairs.

Example 2: Let $A = \{3, 5, 6\}$ and $B = \{7, 9, 14, 15, 18\}$. The pair $(\mathcal{X}, \mathcal{Y})$ is *not* a cqs-pair, although $\mathcal{X}$ and $\mathcal{Y}$ are cqs, respectively, as illustrated in Figure 4.2.

4.1.3 Grid Quorum System Pair: gqs-pair

Now, we introduce another design, *grid quorum system pair* (gqs-pair) of heterogenous quorum systems.

Definition 9 ***Grid quorum system pair (gqs-pair).*** *If a quorum in a grid quorum system contains one row and one column of elements, the* gqs-pair *is a pair consisting of any two qps.*

Lemma 2 *The* gqs-pair *satisfies the heterogeneous rotation closure property and can be a solution to the h-QPS problem.*

Proof 4 *It has been proven in [30] that the grid quorum system satisfies the rotation closure property. Thus, we only need to prove that for two grid quorum systems $\mathcal{X}$ over $\{0, \cdots, n-1\}$ and $\mathcal{Y}$ over $\{0, \cdots, m-1\}$ ($n \leq m$, $p = \lceil \frac{m}{n} \rceil$), $\forall G^p \in \mathcal{X}^p, H \in \mathcal{Y}$, $i \in \{0, \cdots M-1\}$, there is $G^p \cap (H+i) \neq \emptyset$ or $(G+i)^p \cap H \neq \emptyset$.*

Consider a quorum G from $\mathcal{X}$ which contains all elements in the column c, namely c,$c+\sqrt{n}$,$\cdots$,$c+\sqrt{n}(\sqrt{n}-1)$, where $0 \leq c < \sqrt{n}$. Then, a quorum $(G+i)^p$ from the $p-extension$ of $\mathcal{X}$ contains elements, which has the largest distance of $\sqrt{n}$ between any two consecutive elements. $(G+i)^p$ must have an intersection with H since H contains a full row which has $\sqrt{m}$ ($\geq \sqrt{n}$) consecutive integers. Thus, the grid quorum system pair *satisfies the heterogeneous rotation closure property and can be a solution to the h-QPS problem.*

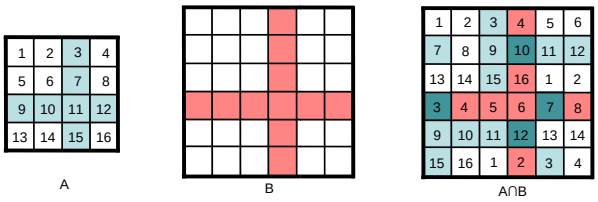


Figure 4.3: An example grid quorum system pair and its rotation closure property: grid quorum system A has a grid 4×4 and B has a grid 6×6. A quorum from A and a quorum from B overlap at 3 slots with B's cycle length.

An illustration on the heterogeneous rotation closure property of the gqs-pair is given in Figure 4.3. There are two grid quorum systems in Figure 4.3, A with the size of 4×4 and B with the size of 6×6. Without considering clock drift, we can see that A's quorums will intersect with B's quorums in the 10^{th}, 3^{rd}, 7^{th}, and the 12^{th} slot.

4.1.4 Construction Scheme for cqs-pair

It is straightforward to construct a *gqs-pair* since it contains two arbitrary grid quorum systems. Therefore, we only discuss the construction of cqs-pair, which is non-trivial. In previous works,

exhaustive search has been used to find an optimal solution for the cyclic quorum design [45]. This is not practical when cycle length (n) is large. In this section, we first present a fast construction scheme for cyclic quorum systems and then apply it to the design of a cqs-pair.

4.1.4.1 Multiplier Theorem

We introduce a few concepts to facilitate our presentation.

Definition 10 *Let D be a (v, k, λ)-difference set in an Abelian group $(G, +)$ of order v. For an integer m, we define*

$$mD = \{mx : x \in D\}$$

Then, m is called a multiplier of D if $mD = D + g$ for some $g \in G$. Also, we say that D is fixed by the multiplier m if $mD = D$.

Example: The set $D = \{0, 1, 5, 11\}$ is a $(13, 4, 1)$-difference set in $(\mathbb{Z}_{13}, +)$. Then, $3D = \{0, 2, 3, 7\} = D + 2$, and hence 3 is a multiplier of D.

Definition 11 ***Automorphism.*** *Suppose $(X, \mathcal{A})$ is a design. A transform function α is an automorphism of $(X, \mathcal{A})$ if*

$$[\{\alpha(x) : x \in A\} : A \in \mathcal{A}] = \mathcal{A}$$

.

Definition 12 ***Disjoint cycle representation:*** *The disjoint cycle representation of a set X is a group of disjoint cycles in which each cycle has the form $(x\ \alpha(x)\ \alpha(\alpha(x)) \cdots)$ for some $x \in X$.*

Suppose the automorphism is $x \mapsto 2x\ mod\ 7$. The disjoint cycle representation of $\mathbb{Z}_7$ is as follows: (0) (1 2 4) (3 6 5).

Theorem 7 ***(Multiplier Theorem).*** *Suppose there exists a (v, k, λ)-difference set D. Suppose also that the following four conditions are satisfied:*

1. *p is* prime*;*
2. *$gcd(p, v) = 1$;*
3. *$k - \lambda \equiv 0\ (mod\ p)$; and*
4. *$p > \lambda$.*

Then p is a multiplier of D.

Theorem 8 *Suppose that m is a multiplier of a (v, k, λ)-difference set D in an Abelian group $(G, +)$ of order v. Then there exists a translate of D that is fixed by the multiplier m.*

The proofs of Theorem 7 and Theorem 8 are given in [44]. According to the Theorem of Singer Difference Set, there must exist a $(q^2+q+1, q+1, 1)$-difference set when q is a prime power. Thus, we only consider the $(q^2+q+1, q+1, 1)$-design, where q is a prime power, in our construction scheme.

In the following, we first give an example to illustrate the application of the Multiplier Theorem for the construction of difference sets.

Example. We use the Multiplier Theorem to find a $(21, 5, 1)$-difference set. Observe that $p = 2$ satisfies the conditions of Theorem 7. Hence 2 is a multiplier of any such difference set. By Theorem 8, we can assume that there exists a $(21, 5, 1)$-difference set in $(\mathbb{Z}_{21}, +)$ that is fixed by the multiplier 2. Therefore, the *automorphism* is $\alpha(x) \mapsto 2x \ mod\ 21$. Thus, we obtain the disjoint cycle representation of the permutation defined by $\alpha(x)$ of $\mathbb{Z}_{21}$ as follows:

$$(0)\ (1\ 2\ 4\ 8\ 16\ 11)\ (3\ 6\ 12)\ (5\ 10\ 20\ 19\ 17\ 13)\ (7\ 14)\ (9\ 18\ 15)$$

The difference set we are looking for must consist of a union of cycles in the list above. Since the difference set has a size five, it must be the union of one cycle of length two and one cycle of length three. There are two possible ways to do this, both of which happen to produce the difference set:

$$(3\ 6\ 7\ 12\ 14)\ and\ \ (7\ 9\ 14\ 15\ 18)$$

With the Multiplier Theorem, we can quickly construct $(q^2+q+1, q+1, 1)$-difference sets, where q is a prime power. This mechanism significantly improves the speed of finding the optimal solution relative to the exhaustive method in [45].

After obtaining the difference sets, we use Theorem 5 to build a cqs-pair.

4.1.4.2 Verification Matrix

Armed with Theorem 5, we adopt a verification matrix to check the non-empty intersection property of two heterogeneous difference sets.

Suppose that $A = \{a_1, a_2, \cdots, a_k\}$ in $(\mathbb{Z}_N, +)$ and $B = \{b_1, b_2, \cdots, b_l\}$ in $(\mathbb{Z}_M, +)$ where $N \leq M$ and $p = \lceil \frac{M}{N} \rceil$. The verification matrix is defined as a $pk \times l$ matrix $\mathcal{M}_{l \times pk}$ whose element $m_{i,j}$ is equal to $(b_i - a_j^p)$ mod M, where $a_j^p \in A^p$, as shown below:

$$\mathcal{M}_{l \times pk} = \begin{bmatrix} b_1 - a_1^p & \cdots & b_1 - a_{pk}^p \\ \vdots & b_i - a_j^p & \cdots \\ b_l - a_1^p & \cdots & b_l - a_{pk}^p \end{bmatrix}$$

We can check whether (A, B) is a *heterogeneous cyclic coterie pair* by checking whether $\mathcal{M}_{l \times pk}$ contains all elements from 0 to $M - 1$ or not. If the checking result is true, it means that:

$$\forall d \in \{0, \cdots, M-1\},\ \exists b_i \in B\ and\ a_j^p \in A^p,\ b_i - a_j^p \equiv\ d\ (mod\ M).$$

This indicates that (A, B) is a *heterogeneous cyclic coterie* based on Theorem 5. Otherwise, (A, B) is *not* a *heterogeneous cyclic coterie*. (An example of the verification matrix will be shown in Section 4.1.4.4.)

If two quorum systems $C(A_N, \mathbb{Z}_N))$ and $C(B_M, \mathbb{Z}_M))$ are cyclic quorum systems, respectively, we can verify whether the pair $[C(A_N, \mathbb{Z}_N)), C(B_M, \mathbb{Z}_M)]$ is a cqs-pair by checking whether or not the *verification matrix* constructed from A and B contains all elements from 0 to $M - 1$.

4.1.4.3 Construction Algorithm

In our proposed algorithm for constructing a cqs-pair, we only consider cyclic quorum systems with a cycle length of $(q^2 + q + 1, q + 1, 1)$, where q is a prime power. This is because, we can prove that when q is a prime power, there must exist a $(q^2 + q + 1, q + 1, 1)$-difference set in $(\mathbb{Z}_{q^2+q+1,q+1,1}, +)$ [44].

We describe our algorithm for constructing a cqs-pair at a high-level of abstraction in Algorithm 1. The input of the algorithm is two numbers n and m, which satisfy $n = q^2+q+1$ and $m = r^2+r+1$ and where q and r are prime powers.

By employing our construction algorithm, for two different integers n and m that satisfy $n = q^2 + q + 1$ and $m = r^2 + r + 1$ (q and r being two prime powers, $n \leq m$), it will take $O(n^2)$ and $O(m^2)$ time to build the *disjoint cycle representations*, respectively. After that, the algorithm will check $u \times v \times l \times pk \approx uvm^{3/2}n^{-1/2}$ elements, since $l \approx \sqrt{m}$ and $k \approx \sqrt{n}$, where u and v are numbers of $(n, k, 1)$-difference sets and $(m, l, 1)$-difference sets, respectively. Thus, the total time complexity is $O(uvm^{3/2}n^{-1/2} + m^2)$ for constructing a cqs-pair with input parameters n and m ($n \leq m$).

4.1.4.4 A Complete Application Example

As an example, consider $n = 7$ and $m = 21$. By the Multiplier Theorem, we can easily obtain two $(7, 3, 1)$-difference sets $\{1, 2, 4\}$ and $\{3, 6, 5\}$ in $(\mathbb{Z}_7, +)$. Similarly, there are two $(21, 5, 1)$-difference sets, $\{3, 6, 7, 12, 14\}$ and $\{7, 9, 14, 15, 18\}$ in $(\mathbb{Z}_{21}, +)$. Let $A_7 = \{1, 2, 4\}$, $B_7 = \{3, 6, 5\}$, $A_{21} = \{3, 6, 7, 12, 14\}$, and $B_{21} = \{7, 9, 14, 15, 18\}$.

Totally, there are four pairs of $(7, 3, 1)$-difference sets and $(21, 5, 1)$-difference sets. First, we check the pair $(C(A_7, \mathbb{Z}_7), C(A_{21}, \mathbb{Z}_{21}))$. The constructed verification matrix is as follows:

$$\begin{bmatrix} 2 & 1 & 20 & 16 & 15 & 13 & 9 & 8 & 6 \\ 5 & 4 & 2 & 19 & 18 & 16 & 12 & 11 & 9 \\ 6 & 5 & 3 & 20 & 19 & 17 & 13 & 12 & 10 \\ 11 & 10 & 8 & 4 & 3 & 1 & 18 & 17 & 15 \\ 13 & 12 & 10 & 6 & 5 & 3 & 20 & 19 & 17 \end{bmatrix}$$

Algorithm 1 Constructing cqs-pair

Require: $n = q^2 + q + 1$ and $m = r^2 + r + 1$, q, r are prime powers
  $n \leftarrow q^2 + q + 1$
  $m \leftarrow r^2 + r + 1$
  $p_a \leftarrow$ Multiplier of $(n, k, 1)$-difference set
  $p_b \leftarrow$ Multiplier of $(m, l, 1)$-difference set
  $\alpha_n(x) \leftarrow p_a \cdot x \ (mod\ n)$
  $\alpha_m(x) \leftarrow p_b \cdot x \ (mod\ m)$
  Construct the disjoint cycle representation for $\mathbb{Z}_n$ with $\alpha_n(x)$
  Construct the disjoint cycle representation for $\mathbb{Z}_m$ with $\alpha_m(x)$
  $u \leftarrow$#Num of unions of disjoint cycle being $(n, k, 1)$-difference set
  $\{A_1, \cdots, A_u\} \leftarrow$ the set of unions of disjoint cycles being $(n, k, 1)$-difference set
  $v \leftarrow$#Num of unions of disjoint cycle being $(m, l, 1)$-difference set
  $\{B_1, \cdots, B_v\} \leftarrow$ the set of unions of disjoint cycles being $(m, l, 1)$-difference set
  for i = 1 to u **do**
    for j = 1 to v **do**
      $M_{i,j} \leftarrow$ verification matrix (A_i, B_j)
      $\mathcal{X}_i \leftarrow C(A_i, \mathbb{Z}_n)$
      $\mathcal{Y}_j \leftarrow C(B_j, \mathbb{Z}_m)$
      if $M_{i,j}$ contains all elements from 0 to $m - 1$ **then**
        $(\mathcal{X}_i, \mathcal{Y}_j)$ is a cqs-pair
      else
        $(\mathcal{X}_i, \mathcal{Y}_j)$ is not a cqs-pair
      end if
    end for
  end for

We find that 7 and 14 are not in the matrix. Thus, the pair $(C(A_7, \mathbb{Z}_7), C(A_{21}, \mathbb{Z}_{21}))$ is *not* a cqs-pair. Similarly, we can check that $(C(B_7, \mathbb{Z}_7), C(B_{21}, \mathbb{Z}_{21}))$ is *not* a cqs-pair. But $(C(A_7, \mathbb{Z}_7), C(B_{21}, \mathbb{Z}_{21}))$ and $(C(B_7, \mathbb{Z}_7), C(A_{21}, \mathbb{Z}_{21}))$ are cqs-pairs, respectively.

The cqs-pair can be applied to WSNs for dynamically changing the quorum system (i.e., the cycle length) at each node, in order to meet end-to-end delay constraints and without loosing network connectivity. Table 4.1 shows the available pairs for cycle lengths ≤ 21.

4.1.5 Performance Analysis

4.1.5.1 Average Discovery Delay

We denote the *average discovery delay* as the time between data arrival and discovery of the adjacent receiver (i.e., the simultaneous wake-up of two nodes). Note that this metric does not include

Table 4.1: cqs-pair (for $n, m \leq 21$)

cycle length	7 $A_7 = \{1, 2, 4\}$ $B_7 = \{3, 5, 6\}$	13 $A_{13} = \{0, 1, 3, 9\}$ $B_{13} = \{0, 2, 6, 5\}$ $C_{13} = \{0, 4, 12, 10\}$ $D_{13} = \{0, 7, 8, 11\}$	21 $A_{21} = \{3, 6, 7, 12, 14\}$ $B_{21} = \{7, 9, 14, 15, 18\}$
7	$(C(A_7, \mathbb{Z}_7), C(A_7, \mathbb{Z}_7))$ $(C(B_7, \mathbb{Z}_7), C(B_7, \mathbb{Z}_7))$	$(C(A_7, \mathbb{Z}_7), C(A_{13}, \mathbb{Z}_{13}))$ $(C(A_7, \mathbb{Z}_7), C(B_{13}, \mathbb{Z}_{13}))$ $(C(B_7, \mathbb{Z}_7), C(C_{13}, \mathbb{Z}_{13}))$ $(C(B_7, \mathbb{Z}_7), C(D_{13}, \mathbb{Z}_{13}))$	$(C(A_7, \mathbb{Z}_7), C(B_{21}, \mathbb{Z}_{21}))$ $(C(B_7, \mathbb{Z}_7), C(A_{21}, \mathbb{Z}_{21}))$
13		$(C(A_{13}, \mathbb{Z}_{13}), C(A_{13}, \mathbb{Z}_{13}))$ $(C(B_{13}, \mathbb{Z}_{13}), C(B_{13}, \mathbb{Z}_{13}))$ $(C(C_{13}, \mathbb{Z}_{13}), C(C_{13}, \mathbb{Z}_{13}))$ $(C(D_{13}, \mathbb{Z}_{13}), C(D_{13}, \mathbb{Z}_{13}))$	$(C(B_{13}, \mathbb{Z}_{13}), C(A_{21}, \mathbb{Z}_{21}))$

the time for delivering a message.

Suppose that the length of one time slot is 1.

Theorem 9 *The average discovery delay between two nodes that wakeup based on quorums from the same* cyclic quorum system *adopting the* $(n, k, 1)$*-difference set is:*

$$E(Delay) = \frac{n-1}{2}.$$

Proof 5 *Let the* k *elements in* $(n, k, 1)$*-difference set be denoted as* a_1*,* a_2*,* $\cdots$*,* a_k*. If a node has a message that arrived during the* i^{th} *time slot, the expected delay (from data arrival to the simultaneous wake-up of two nodes) is* $\frac{1}{k}(a_1 - i) \ mod \ n + \frac{1}{k}(a_2 - i) \ mod \ n + \cdots + \frac{1}{k}(a_k - i) \ mod \ n$*. If a message has arrived, the probability of the message arriving during the* i^{th} *time slot is* $\frac{1}{n}$*. Thus, the expected delay (average delay) is:*

$$\begin{aligned}
&E(Delay) \\
&= \frac{1}{n}[\frac{1}{k}(a_1 - 1) \ \% \ n + \frac{1}{k}(a_2 - 1) \% \ n + \cdots + \frac{1}{k}(a_k - 1) \ \% \ n \\
&+ \frac{1}{k}(a_1 - 2) \ \% \ n + \frac{1}{k}(a_2 - 2) \ \% \ n + \cdots + \frac{1}{k}(a_k - 2) \ \% \ n \\
&+ \quad \cdots\cdots \\
&+ \frac{1}{k}(a_1 - n) \ \% \ n + \frac{1}{k}(a_2 - n) \ \% \ n + \cdots + \frac{1}{k}(a_k - n) \ \% \ n] \\
&= \frac{1}{nk} \cdot (k \cdot 1 + k \cdot 2 + \cdots + k \cdot n - 1) \\
&= \frac{n-1}{2}
\end{aligned}$$

Corollary 2 *The average discovery delay between two nodes that wakeup based on a cqs-pair in which two* cyclic quorum systems *have cycle lengths n and m ($n \leq m$), respectively, is:*

$$\frac{n-1}{2} < E(Delay) < \frac{m-1}{2}$$

Corollary 2 indicates that the average discovery delay between two nodes that adopt a cqs-pair is bounded. When the average one-hop delay constraint is D, we must meet $\frac{m-1}{2} \leq D$.

Theorem 10 *The average discovery delay between two nodes that wakeup based on quorums from the same* grid quorum system *with a grid of $\sqrt{N} \times \sqrt{N}$ elements is:*

$$E(Delay) = \frac{(N-1)(\sqrt{N}+1)}{3\sqrt{N}}$$

The detail proof is presented in [46].

Corollary 3 *The average discovery delay between two nodes that wakeup based on a gqs-pair in which two grid quorum systems adopt a grid of $\sqrt{n} \times \sqrt{n}$ and a grid of $\sqrt{m} \times \sqrt{m}$, respectively, is:*

$$\frac{(n-1)(\sqrt{n}+1)}{3\sqrt{n}} < E(Delay) < \frac{(m-1)(\sqrt{m}+1)}{3\sqrt{m}}$$

We omit the proof for Corollary 3 due to the straightforwardness.

4.1.5.2 Optimal Quorum Ratio and Energy Conservation

We define **quorum ratio**, denoted ϕ, as the proportion of the beacon intervals that is required to be awake in each cycle. Correspondingly, the energy conservation ratio of a node is $1 - \phi$.

As claimed in [45], Cqs is an optimal design where the optimality means given a schedule with cycle length n, cqs design has the minimum quorum size k to make sure there is always not-empty intersection between this schedule and any rotations of the schedule. This is not difficult to explain. As discussed in Section 4.1.4, a cqs design is based on $(q^2+q+1, q+1, 1)$-difference set which means given a qualified set $A = \{a_1, ..., a_k\}(\mod N)$, there is *exactly* one ordered pairs (a_i, a_j), $a_i, a_j \in A$ such that $a_i - a_j \equiv d\ (modN)$ for every $d \neq 0$. Any reduction of elements in A will lead that $a_i - a_j \equiv d\ (modN)$ for every $d \neq 0$ cannot be met. Thus, given a cycle length n where $n = q^2+q+1$ and q is a prime power, for a cqs schedule which is based on a $(n, k, 1)$-difference set, its quorum ratio is the minimum one among all possible designs.

We restrain to the case of $n = q^2+q+1$ is because the authors in [44] have proved that a $(q^2+q+1, q+1, 1)$-difference set exists and that the optimal quorum ratio is $\phi = \frac{q+1}{q^2+q+1}$ for such a cyclic quorum system.

For a cqs-pair, the *quorum ratios* for systems in the pair which are based on (N, k, M, l)-difference pair are:

$$\phi_1 = \frac{\sqrt{4N-3}+1}{2N} \; and \; \phi_2 = \frac{\sqrt{4M-3}+1}{2M}$$

respectively, where $n = q^2 + q + 1$ and q is a prime power. Since cqs has optimal quorums ratio, the two systems in the cqs-pair have optimal quorum ratio respectively.

For a grid quorum system with $\sqrt{n} \times \sqrt{n}$ grid, the *quorum ratio* is:

$$\phi = \frac{2\sqrt{n}-1}{n}$$

and the corresponding energy saving ratio is:

$$1 - \phi = 1 - \frac{2\sqrt{n}-1}{n}$$

Recalling the average discovery delay in Section 4.1.5.1, we can observe that there is a trade-off between the average delay and the quorum ratio. Larger the cycle length of a quorum system, larger is the discovery delay, but smaller is the quorum ratio.

4.1.6 Simulation Results

We evaluated the performance of our schemes through numerical studies and by real implementation over a WSN platform of Telosb motes [10]. In our experiments, a set of nodes was deployed. The radio range was configured to 10 meters for each node. There was one sink node which acted as the base station. The sink node communicated with a laptop computer (through a wireless USB serial port), which recorded performance measurements. The detailed radio parameters such as data rates default to the data sheet of TelosB [10].

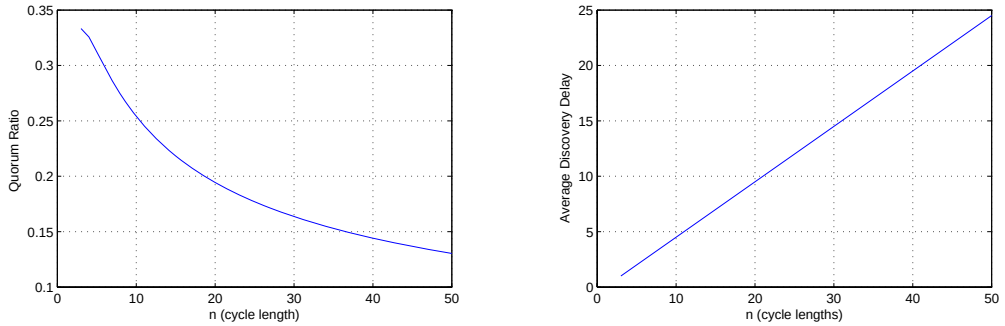
We built our wakeup scheduling schemes over the basic CSMA/CA protocol. We used MintRoute [91] as the routing protocol for end-to-end transmission. Traffic load was generated by a Poisson distribution [92] with rates in the range 10-100 packets/sec. Each packet only contains one Active Message whose size is defined in TinyOS 2.0 [93].

Two important performance metrics were measured in our experimental study: (1) quorum ratio and energy saving ratio; and (2) average neighbor discovery delay.

4.1.6.1 Performance Trade-off

We first evaluated the quorum ratio and average neighbor discovery delay by numerical analysis.

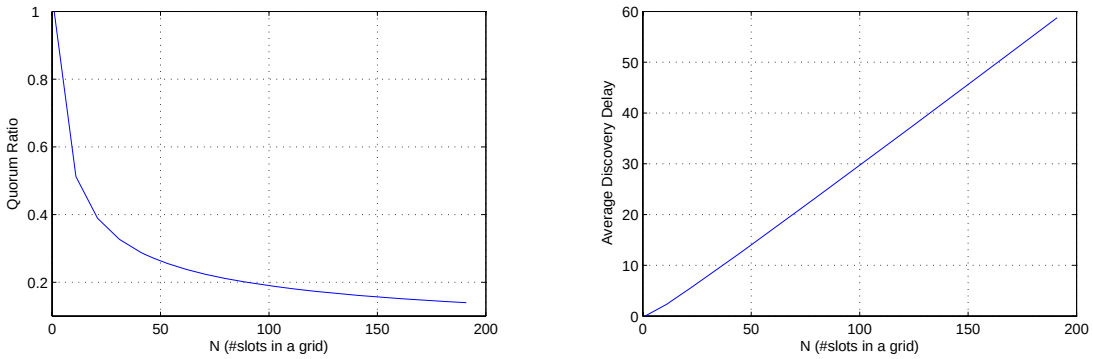
The performance of a *cyclic quorum system* is shown in Figure 4.4. There is a trade-off between quorum ratio and average discovery delay since they have reverse changing trends under increasing cycle lengths.



(a) Quorum Ratio (b) Average Discovery Delay

Figure 4.4: Quorum Ratio and Average Discovery Delay for Cyclic Quorum Systems (Numerical Results)

The performance of a *grid quorum system* is shown in Figure 4.15. The grid quorum system's quorum ratio is bigger than that of the cyclic quorum system with identical cycle length, but the average discovery delay is approximately $2/3$ of that of the corresponding cyclic quorum system.



(a) Quorum Ratio (b) Average Discovery Delay

Figure 4.5: Quorum Ratio and Average Discovery Delay for Grid Quorum Systems (Numerical Results)

4.1.6.2 Impact of Heterogeneity

For heterogenous quorum-based wakeup scheduling, like cqs-pair or gqs-pair, the cycle lengths of two quorum systems are different. We evaluated the impact of heterogeneity of two different cycle lengths on the average discovery delay between two neighbor nodes.

For this set of experiment, we focused on the cqs-pair since it is an optimal design. We fixed the traffic load between two nodes at $10\ packets/sec$ in the experiment.

We varied the cycle lengths of two neighbor nodes in two (different) quorum systems in a cqs-pair. The cycle length of one node was varied from 7 to 58. The neighbor node, which used a counterpart cyclic quorum system, had its cycle length varied from 7 to 21. We do not show the impact on the energy consumption ratio when the cycle lengths of cqs-pair were varied, since the energy consumption ratio of a node is mainly dependent upon its own cycle length, which has already been evaluated in Section 4.1.6.1.

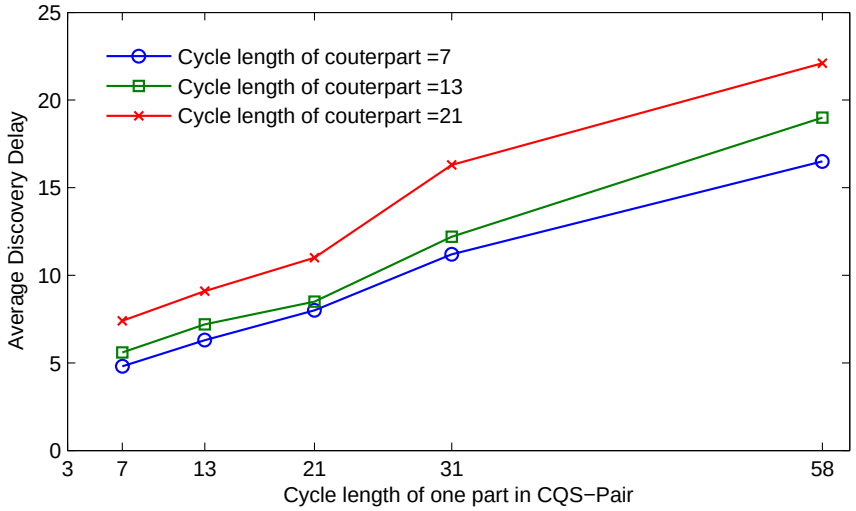


Figure 4.6: Impact of Heterogeneity

Figure 4.6 shows how the average discovery delay changes with different cqs-pairs. When one part in a pair keeps its cycle length constant and the counterpart increases its cycle length, the average discovery delay between them almost increases linearly.

4.1.6.3 Impact of traffic load

In this section, we report our experiments on measuring the impact of traffic load on the performance of cqs-pair and comparisons with the basic CSMA/CA MAC protocol. We varied the traffic load from 10 packets/sec to 100 packets/sec in the experiments. The cycle lengths of cyclic quorum systems in the cqs-pair were chosen among 7, 13 and 21.

Figure 4.7 shows how the energy consumption ratios of nodes adopting different cyclic quorum systems increase under increasing traffic load between two neighboring nodes. The rationale is that higher traffic loads will cause a node to increase its wakeup time ratio in our implementation. When the traffic load is low, the impact is insignificant because a node will maintain its current wakeup schedule, without adding more wakeup slots into its communication schedule for transmitting or for receiving packets.

Figure 4.8 shows that the average discovery delay decreases with the increasing of traffic load. This

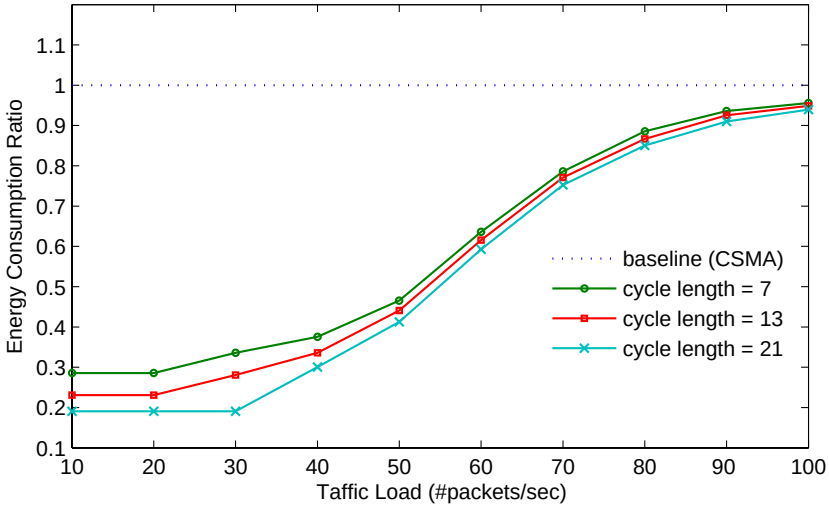


Figure 4.7: Impact of Traffic Load: Energy Consumption Ratio

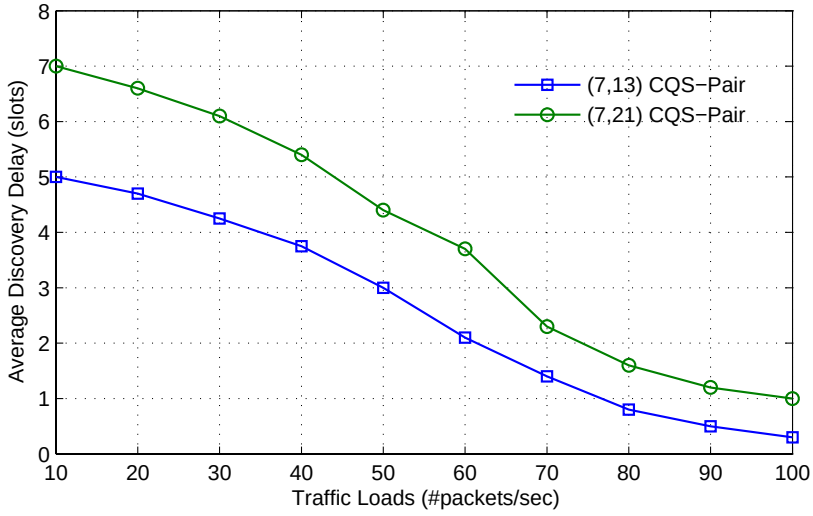


Figure 4.8: Impact of Traffic Load: average discovery delay

is because, the communication schedule of a node will have more active slots, when compared with its quorum-based wakeup schedule during high traffic load. With more slots staying awake, the average discovery delay between two neighboring nodes will be significantly reduced.

4.1.7 Conclusions

We presented a theoretical approach for heterogeneous asynchronous wakeup scheduling in WSNs. We first defined the h-QPS problem—i.e., given two cycle lengths n and m ($n < m$), how to design a pair of heterogeneous quorum systems to guarantee that two adjacent nodes that select heterogenous quorums from the pair as their wakeup schedules can hear each other at least once in every m consecutive time slots. We proposed two designs for heterogeneous asynchronous wakeup scheduling: the cyclic quorum system pair (cqs-pair) and the grid quorum system pair (gqs-pair). We also presented a fast construction scheme to assemble a cqs-pair. In our construction scheme, we first quickly construct an $(n, k, 1)$-difference set and an $(m, l, 1)$-difference set. Based on two difference sets A in $(\mathbb{Z}_n, +)$ and B in $(\mathbb{Z}_m, +)$, we can construct a cqs-pair ($C(A, \mathbb{Z}_n), C(B, \mathbb{Z}_m)$) when A and B can form a (n, k, m, l)-difference pair.

The performance of a cqs-pair and a gqs-pair were analyzed in terms of average delay, quorum ratio, and energy saving ratio. We show that the average delay between two nodes that wakeup via heterogenous quorums from a cqs-pair is bounded between $\frac{n-1}{2}$ and $\frac{m-1}{2}$, and the quorum ratios of the two quorum systems in the pair are optimal, respectively, given their cycle lengths n and m. For a gqs-pair with $\sqrt{n} \times \sqrt{n}$ grid and $\sqrt{m} \times \sqrt{m}$ grid, the average discovery delay is bounded within $\frac{(n-1)(\sqrt{n}+1)}{3\sqrt{n}} < E(Delay) < \frac{(m-1)(\sqrt{m}+1)}{3\sqrt{m}}$, while the quorum ratios are $\frac{2\sqrt{n}-1}{n}$ and $\frac{2\sqrt{m}-1}{m}$, respectively.

4.2 Asymmetric Quorum-based Wakeup Scheduling

4.2.1 Motivations

Even though some works introduce adaptive configuration mechanism, like B-MAC [13] and T-MAC [24], for dynamically changing duration of time slots, these solutions may suffer energy waste in neighbor discovery over unreliable environment with link or node failures. The reason is as follows. When a node send out preamble to detect one neighbor in a time slot, it probably does not receive an ACK (i.e., to indicate the presence) message from the neighbor within that time slot; but the node cannot distinguish two possible causes of not receiving ACK: it may be because the neighbor has longer time slot so that the neighbor is still sleeping, or because of unreliable environment which leads to message loss. Since the probing node cannot distinguish the two possible reasons, it has to continuously send out preamble to probe its neighbor, which will cause huge energy waste if it takes a long time to recover the failed links. If the probing nodes use

some back-off mechanisms (i.e., exponential back-off), the neighbor discovery latency cannot be bounded.

Motivated by supporting energy-efficient neighbor discovery over unreliable environment and by supporting asymmetric energy saving in idle state, we present p-Grid, an asynchronous neighbor discovery protocol for low duty-cycled WSNs. In p-Grid, each node listens the channel states only in the beginning of selected time slots in every n consecutive time slots. Here, n is the cycle length and the selected time slots are referred as to a quorum schedule. When a node wants to discover its neighbor, it stays awake and sends out preambles to probe its neighbors in selected time slots which are also referred as to a quorum schedule. Our design guarantees that the quorum-based probing schedule of a node will intersect at least once with the listening schedules of its neighbors within bounded time slots.

In p-Grid, the asymmetric energy saving ratio is achieved by adopting asymmetric quorum-based probing scheduling and listening scheduling via grid quorum group design. By theoretical analysis, we show that p-Grid achieves flexibility for run-time configuration by selecting different quorums for listening and neighbor probing. It is also shown that p-Grid wastes less energy for detecting neighbors over unreliable environment, and can guarantee neighbor discovery within bounded latency once the failed links are recovered.

The main objective in designing p-Grid is to support flexible run-time configuration of idle energy saving ratio and to achieve energy efficiency for neighbor discovery over unreliable environment. In addition, the protocol should be asynchronous for easy implementation and be easily integrated into existed MAC protocols or routing protocols. The following problems are involved in our design: (1) how to designs an energy-efficient listening schedule and node probing schedule, rather than a simple periodic listening and probing as that in B-MAC or X-MAC; and (2) how to guarantee bounded neighbor discovery latency with non-periodic schedules; and (3) what is an energy-efficient back-off mechanism in case of node failure over unreliable environment?

4.2.2 Protocol Design of p-Grid

4.2.2.1 Overview

The design of p-Grid is based on properties of quorum groups. A quorum group contains two types of quorums: read quorum and write quorum which satisfy $write \cap write \neq \emptyset$ and $read \cap write \neq \emptyset$.

There are two operations modes for each node in our design: listening mode and probing mode. In listening mode, a node checks the gain level of its channels to detect whether there is a message coming, in the beginning of selected time slots. In probing mode, a node will stay awake and sends out short strobe preambles which is referred as to DISC messages in selected time slots. Every node stay in listening mode in idle state. For neighbor discovery, the probing node simply switches from listening mode to probing mode, and then sends out preambles to probe its neighbors. When the neighbors detect the preambles, they will feedback REPLY messages to indicate their presence.

In p-Grid, the listening schedules of nodes in listening mode are complying with a read quorum. The probing schedules of nodes in probing mode are based on a write quorum. Since $read \cap write \neq \emptyset$, a node in probing mode can definitely discover its neighbors in listening mode.

In order to achieve flexibility in run-time configuration of duty cycle, a node only needs to select a schedule with different size, rather than changing the duration of each time slot. The bounded latency is achieved by non-empty intersection property among read quorums and write quorums within one cycle of the quorum group. We will also show that our quorum-based schedules can act as good back-off schemes over unreliable environment.

4.2.2.2 Listening Mode and Probing Mode

We introduce the operations of nodes in listening mode and probing mode. The operations have some similarities with previous duty-cycled asynchronous MAC protocols, like B-MAC [13] or X-MAC [27].

In our design, for nodes in listening mode, they check the channel states in the beginning of time slots which are selected as read quorum that is defined in Definition 13. Let R_j denote the read quorum which contains the j^{th} column of entries in the $n \times n$ array. A node checks the channel state by dual preamble sampling in two separate periods P_1 and P_2 in each slot of R_j. Figure 4.9(a) shows an example. Here, the channel state is checked in the beginning of blue colored slots. If the gain of the channel is less than a threshold, the node returns to sleep state after checking. We choose the dual preamble sampling in order to further reduce the awake time in listening mode.

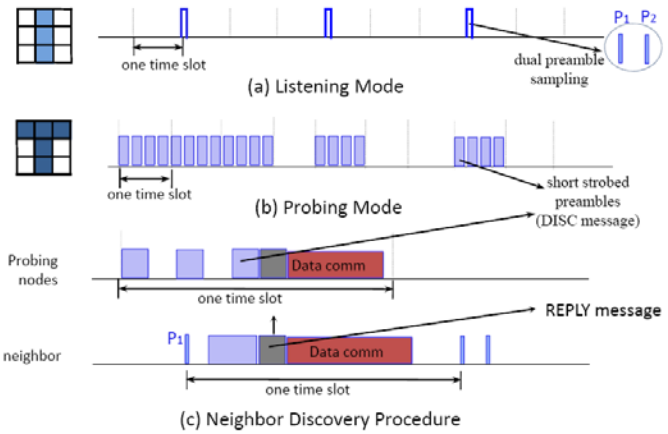


Figure 4.9: Two operation modes and neighbor discovery procedure in p-Grid

In the probing mode, a node sends out a series of short strobe preambles in time slots which are

selected as write quorum that is defined in Definition 14. We denote the short preamble message as DISC. Let $W_{j,i}$ denote the write quorum which contains the j^{th} column of entries and the i^{th} row of entries in the $n \times n$ array. Figure 4.9(b) shows an example for probing mode.

The neighbor discovery procedure, as shown in Figure 4.9(c), is as follows. A neighbor node checks its channel states during P_1 and P_2. If the channel is not free during P_1, the neighbor node enters the sleep mode for a period of time and proceeds to receive DISC messages. The period P_1 is necessary only when the channel is found to be free during P_1. After receiving a DISC message, the node will immediately feedback an acknowledgement message, referred as to REPLY message, to the probing node, and will keep awake for one slot to receive any possible following messages. Otherwise, the neighbor node will keep staying in listening mode.

At the probing node side, if the probing node does not receive a REPLY message after sending a DISC message, it will continue to send out DISC messages until the current time slot expires. The probing node will continue to send out DISC messages in the next slot of $W_{j,i}$. If the incoming time slot is not in $W_{j,i}$, the probing node will stay in the SLEEP state until the arrival of a next slot belonging to $W_{j,i}$.

We formally define read quorum and write quorum as follows. Suppose the quorum group has a $n \times n$ grid. We arrange $n \times n$ consecutive time slots into a $n \times n$ array.

Definition 13 *[Read quorum] In a $n \times n$ array, we define the read quorum as a set containing any* column *of time slots in the grid. If the read quorum contains the j^{th} column of slots, we denote it as Q_j^r.*

Definition 14 *[Write quorum] In a $n \times n$ array of consecutive time slots, we define the write quorum as a set containing any* row *of time slots in the grid. If the write quorum contains the i^{th} row of slots, we denote it as Q_i^w.*

Definition 15 *[Grid quorum group] In the $n \times n$ array, we define the grid quorum group as the set of all read quorums and write quorums.*

As to the value of R_j and $W_{j,i}$, they can be selected randomly from a grid quorum group. We will show the nonempty intersection property between an arbitrary read quorum and a write quorum in next section. In addition, there is difference between the grid quorum group and grid quorum systems. In grid quorum groups, it is not necessary for two read quorums or two write quorums to intersect with each other. But in grid quorum system, any two quorums have a non-empty intersection.

4.2.2.3 Neighbor Discovery for Symmetric Groups

We first show the non-empty intersection property between read quorums and write quorums which belongs to same grid quorum groups. Such intersection property is the foundation for the neighbor

discovery within bounded latency.

Theorem 11 *With one grid quorum group, a probing node equipped with a write quorum as its probing schedule must discover its neighbors equipped with a read quorum as listening schedules by at least once, regardless of clock drift.*

We omit the proof for Theorem 11. The formal proof is similar as the proof for asynchronous intersection property in [30]. An intuitive example is given in Figure 2. Two nodes adopt a 3×3 grid quorum group. The probing mode selects a write quorum containing the 2nd column plus the 1st row in the grid. The listening mode selects a read quorum containing the 3rd column in the grid. Regardless of clock drift, they have at least one overlapped slot within 3×3 slots. With the help of preamble at probing node side and channel state checking at neighbor side, the probing node can detect the neighbor node in its 2^{nd}, 5^{th} and 8^{th} time slot.

Corollary 4 *Two write quorums in a quorum group must have at least one intersection.*

In order to reduce the discovery latency, it is better to select a write quorum which contains the first row of the grid so that the intersection will happen within the first n slots. We denote such write quorum as $W_{j,1}$ where $j \in [1..n]$.

In the procedure for neighbor discovery, the key step is quorum switch. A probing node has to first switch its listening schedule from a read quorum to a write quorum. The reason for doing quorum switch is that two read quorum cannot guarantee non-empty intersection. Let the read quorum be R_j and the write quorum be $W_{j',1}$. After switching, although there is an increase in the quorum size for write quorum, we have $R_j \cap W_{j',1} \neq \emptyset$ within n^2 time slots.

4.2.2.4 Neighbor Discovery for Asymmetric Groups

In order to satisfy different energy saving requirements for heterogenous nodes, it is desirable that two nodes adopt asymmetric listening scheduling patterns. For example, the listening schedule of the cluster head which has more power supply may use quorums from 3×3 grid quorum groups, while the schedule for cluster members with less power resource uses quorums from 5×5 grid quorum groups. In this section, we present the neighbor discovery scheme in such scenario.

It is difficult to guarantee the non-empty intersection property between read quorums and write quorums from two arbitrary grid quorum groups. However, if we choose prime-grid quorum groups which are defined as follows, the nonempty intersection property can be satisfied.

Definition 16 *In the grid quorum group with $n \times n$ time slots, if n is a* prime*, we define the quorum group as* prime-grid quorum group.

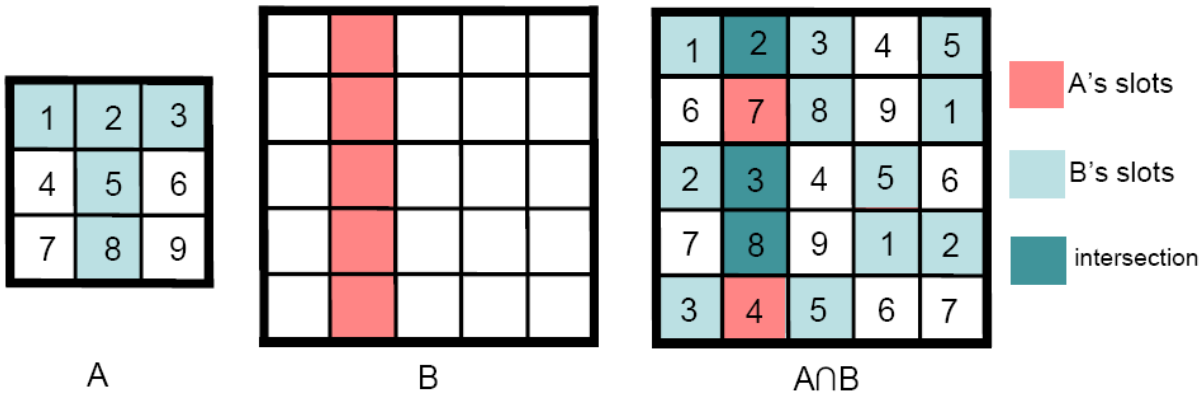


Figure 4.10: Intersection between a write quorum and a read quorum from two prime-grid quorum groups: the write quorum A from the grid quorum group $\mathcal{Q}_{3\times3}$, and the read quorum B from the grid quorum group $\mathcal{Q}_{5\times5}$. $A^2 \bigcap B \neq \emptyset$

Definition 17 *(*p*-extension). Given two positive integers n and p, for a set $A = \{a_i | 1 \leq i \leq k, a_i \in \mathbb{Z}_n\}$, the* p-extension *of A is defined as $A^p = \{a_i + j*n | 1 \leq i \leq k, 0 \leq j \leq p-1, a_i \in \mathbb{Z}_n\}$.*

Example: Let $A = \{1, 2, 3, 4\}$ in $(\mathbb{Z}_{16}, +)$. Now, $A^2 = \{1, 2, 3, 4, 17, 18, 19, 20\}$ in $(\mathbb{Z}_{32}, +)$.

Theorem 12 *Consider two different prime-grid quorum groups $\mathcal{Q}_A$ with $n \times n$ array and $\mathcal{Q}_B$ with $m \times m$ array. Suppose $n < m$ and $p = \lceil \frac{m}{n} \rceil$. The* p-extension *of a read quorum $A \in \mathcal{Q}_A$ must have at least one intersection with a read quorum B from $\mathcal{Q}_B$, or $A^p \cap B \neq \emptyset$.*

Proof 6 *Let $A^p = \{a_1, a_2, \cdots, a_n | a_i < pn^2\}$. All elements in A satisfy $a_i \equiv r_a (mod\ n)$, where r_a is the residue and $r_a < n$.*

Let $B = \{b_1, b_2, \cdots, b_m\}$. All elements in B satisfy $b_i \equiv r_b (mod\ m)$, where r_b is the residue and $r_b < m$.

Since n and m are prime numbers, according to the Chinese Remainder Theorem, there must be an integer I that satisfies $I \equiv r_a (mod\ n) \equiv r_b (mod\ m)$ and $I \leq n \times m < m^2$. This means that there is an intersection between A^p and B.

Lemma 3 (small write to big read) *Consider two prime-grid quorum groups $\mathcal{Q}_A$ with $n \times n$ array and $\mathcal{Q}_B$ with $m \times m$ array. Suppose $n < m$. The* p-extension *($p = \lceil m/n \rceil$) of a write quorum $W_{j,i}$ from $\mathcal{Q}_A$ must have at least* one *intersection with a read quorum $R_{j'}$ from $\mathcal{Q}_B$, or $(W_{j,i})^p \cap B_{j'} \neq \emptyset$.*

Proof 7 *Let the read quorum consisting of the j^{th} column of slots in $\mathcal{Q}_A$ be denoted as RA_j. Since $(RA_j)^p \subset (W_{j,i})^p$ and $(RA_j)^p \cap R_{j'} \neq \emptyset$, based on Theorem 12, we have $(W_{j,i})^p \cap R_{j'} \neq \emptyset$.*

An illustration of Lemma 3 is given in Figure 4.10. There are two grid quorum groups in Figure 4.10, one with a 3×3 grid and another one with a 5×5 grid. We can see that the write quorum A in the smaller group will intersect with the read quorum B of the bigger group at three time slots.

Lemma 4 (big write to small read) *Consider two prime-grid quorum groups $\mathcal{Q}_A$ with $n \times n$ array and $\mathcal{Q}_B$ with $m \times m$ array. Suppose $n < m$ and $p = \lceil \frac{m}{n} \rceil$. A write quorum $W_{j,i}$ from $\mathcal{Q}_B$ must have at least* two *intersections with the* p*-extension of a read quorum $R_{j'}$ from $\mathcal{Q}_A$.*

We omit the proof for Lemma 4 since it is intuitive.

Theorem 13 *Given two adjacent nodes and two prime-grid quorum groups $\mathcal{Q}_{n \times n}$ and $\mathcal{Q}_{m \times m}$, if one node adopts a read quorum from Qnn and the other node adopts a write quorum from $\mathcal{Q}_{m \times m}$, the two nodes can discover each other within $\max\{n^2, m^2\}$ time slots, by operating in listening mode and probing mode separately.*

Proof 8 *Assume that the read quorum is R from $\mathcal{Q}_{n \times n}$ and the write quorum is W from $\mathcal{Q}_{m \times m}$.*

Without loss of generality, assume that the read quorum is Q^r from $\mathcal{Q}_{n \times n}$ and the write quorum is Q^w from $\mathcal{Q}_{m \times m}$.

Case 1: $n = m$. R and W must have an intersection within n^2 slots, according to Theorem 11.

Case 2: $n < m$. $p = \lceil m/n \rceil$. Based on Lemma 4, R^p and W must have at least two intersections within m^2 slots.

Case 3: $n > m$. $p = \lceil n/m \rceil$. Based on Lemma 4, R^p and W must have at least one intersection within n^2 slots.

Since there must be an overlapped slot between R and W, the nodes will discover each other with the help of dual preamble sampling and shorted strobed RTS.

4.2.2.5 Backoff Mechanism in Unreliable Environment

With asymmetric design, when a node does not receive REPLY message after sending out DISC message, it is probably because the neighbor choose a bigger quorum group, or because of link failures. But the probing node cannot distinguish the two possible reasons.

By nature, our design can provide an energy efficient back-off mechanism to avoid periodic retries after not receiving a REPLY message. In p-Grid, the node who wants to discovery others will gradually enlarge the cycle length of the grid quorum group for its probing scheduling when there is no REPLY messages received in one quorum cycle, until timeout. The value of timeout is set in specific application requirement, e.g, in a network with node mobility, the value can be smaller compared to the value in a static network.

Since we adopt prime-grid quorum group design, the back-off mechanism is operated as follows: a node in probing mode chooses a quorum group sized with n_1^2 , if the node does not receive REPLY message after some cycles (i.e. ≥ 3, it is dependent on user configuration), the node chooses a bigger quorum group sized with n_2^2 , if the node still does not receive REPLY message within n_2^2 time slots, the node chooses a bigger group sized with n_3^2. The node will continue to increase the size of next quorum group, until timeout. Suppose the last quorum group before timeout is sized by $n_p \times n_p$, the selecting sequence satisfies,

$$n_1 < n_2 < \cdots < n_p$$

whether n_1, n_2, $\cdots$, n_p are primes.

The rationale for the backoff mechanism is straightforward. With the increasing of quorum group cycle, at some time, the cycle n_k^2 ($1 \leq k \leq p$) will be equal to or bigger than the cycle of neighbors, which can meet the non-empty intersection property within one cycle, according to Lemma 4.

As we will show in Section 4.2.2.2, it is more energy efficient with larger quorum cycles. Therefore, we can achieve energy efficiency in unreliable environment by enlarging the size of selected grid quorum group, but meanwhile guarantee the neighbor discovery.

4.2.3 Performance Analysis

In this section, we show the energy efficiency of p-Grid in unreliable environment and its flexibility in supporting heterogenous energy saving ratio.

For the purpose of easy presentation, we denote Ti as the length of one time slot, T_{awake} as the total time duration for channel sampling(s) and Ts as the time duration for single channel sampling ($T_{awake} = 2T_s$ for dual preamble sampling). Suppose there are two prime grid quorum groups: $\mathcal{Q}_{n\times n}$ and $\mathcal{Q}_{m\times m}$($n < m$, n and m are primes).

The typical value for T_i in our design is set to 50ms. We set the default duration for preamble sampling (T_s) as 3 milliseconds when a node performs a sequence of operations to startup the radio, and set the radio in the RX mode, which is default to the setting in [13]. Thus the total duration for the dual preamble sampling is 6 milliseconds.

4.2.3.1 Flexibility of Configuration

Suppose the grid quorum group is $\mathcal{Q}_{n\times n}$ whose time axes is organized into consecutive slots with a $n \times n$ array.

We define the *idle energy consumption ratio* as the total awake time to the whole operation time for a node. Suppose the duty cycle in a single time slot is $\alpha = \frac{T_{awake}}{T_i}$, where T_{awake} is the duration for channel sampling. In our dual preamble sampling scheme, $T_{awake} = 2T_s$. The quorum ratio, which is the proportion of the size of a quorum in a whole quorum group cycle, is $\frac{1}{n}$ for read quorums.

Thus, the overall *idle energy consumption ratio* for read/write quorums in $\mathcal{Q}_{n\times n}$ in idle state is: $\phi = \frac{1}{n} \cdot \alpha$.

The overall *idle energy consumption ratio* for write quorums in $\mathcal{Q}_{n\times n}$ in idle state is:

$$\phi = \frac{2n-1}{n^2} \cdot \alpha \tag{4.1}$$

To support different idle energy saving requirement, a node only needs to adjust its quorum ratio, $\frac{1}{n}$ for read/write quorums and $\frac{2n-1}{n^2}$ for write quorums. This can be done by selecting different grid quorum group in a node. The convenience of run-time configuration stems from no changes for duty cycle in a singe time slot. The run-time configuration can be implemented in application layer.

4.2.3.2 Energy Efficiency in Unreliable Environment

For XMAC-ALPL, suppose that the cluster-head adopts a checking interval with duration of L_c and the cluster members adopt a checking interval with duration of L_m ($L_c \leq L_m$).

In p-Grid, suppose that the cluster-head adopts a quorum group of $\mathcal{Q}_{n\times n}$ and the cluster members adopt a $\mathcal{Q}_{m\times m}$.

Theorem 14 *In reliable environment with no link failure, the worst case energy consumption of p-Grid for neighbor discovery is same as that of XMAC-ALPL, with respect to same idle energy consumption ratio.*

Proof 9 *Suppose the duration of channel checking in both p-Grid and XMAC-ALPL is T_{awake}. The idle energy consumption ratio of XMAC-ALPL for the cluster-head is T_{awake}/L_i. The idle energy consumption ratio of p-Grid is $\frac{T_{awake}}{T_i * n}$. If the two mechanisms have same idle energy consumption ratio, we have $L_c = T_i * n$. Regarding cluster members, similarly, we have $L_m = T_i * m$.*

Case 1: The cluster member is the sender. For neighbor discovery, the worst cast awake time in XMAC-ALPL will be L_c since the cluster member will definitely discover the cluster head after L_c once it begins to send out preamble.

The worst case awake slots for the cluster members in p-Grid will be n slots before discovering the receiver. The awake time will be $T_i * n$. Since $L_c = T_i * n$, the worst case energy consumption of p-Grid for per neighbor discovery is same as that of XMAC-ALPL.

Case 2: The cluster head is the sender. We can prove the theorem similarly as that in case 1.

Theorem 15 *In unreliable environment with link failures, the worst case energy consumption of p-Grid for neighbor discovery is better than that of XMAC-ALPL, with respect to same idle energy consumption ratio.*

Proof 10 *Case 1: The cluster member is the sender. In unreliable environment, the senders with XMAC-ALPL mechanism have to send out preamble continually. Thus for neighbor discovery, the worst cast awake time of the sender will be $k * L_c$ where the kink will recover in the k^{th} slots after the initial link failure.*

As to p-Grid, the sender will discover the receiver within $\lceil \frac{k*L_c}{m*n*T_i} \rceil$ quorum slots. The awake time is $\lceil \frac{k*L_c}{m*n*T_i} \rceil * T_i$. If $m > 1$ and $n > 1$, we have $\lceil \frac{k*L_c}{m*n*T_i} \rceil * T_i \leq (\frac{k*L_c}{m*n*T_i} + 1) * T_i = (\frac{k}{m*n} + \frac{1}{n}) * L_c \leq k * L_c$.

Case 2: The cluster member is the sender. We can make similar proof.

Thus, p-Grid is energy-efficient than XMAC-ALPL in unreliable environment. In reliable situation, they can achieve same performance regarding energy efficiency as claimed in Theorem 14.

4.2.3.3 Bounded Neighbor Discovery Latency

Discovery latency of homogenous case: We define the *discovery latency* as the time between the data arrival and the discovery of the adjacent receiver (i.e., receiving the CTS message from the receiver). Note that this metric does not include the time for delivering a message.

Theorem 16 *The average discovery latency between two nodes that use a read quorum and a write quorum, respectively, from the same grid quorum group $\mathcal{Q}_{n\times n}$ as their LPL schedules is $D = \frac{n}{2} \cdot T_i$.*

Proof 11 *The latency can be divided into two parts: 1) the latency for finding the first intersection, denoted as D_1; and 2) the latency to hear from the receiver within the intersected slot, denoted as D_2.*

For the first part, suppose the write quorum is $Q^w_{j,1}$ and the read quorum is $Q^r_{j'}$. The probability of $Q^r_{j'}$ overlapping with $Q^w_{j,1}$ in the k^{th} ($1 \leq k \leq n$) slot of $Q^w_{j,1}$ is $1/n$, with corresponding latency $(k-1) \cdot T_i$. Thus,

$$D_1 = \frac{1}{n} \sum_{k=1}^{n} (k-1) \cdot T_i = \frac{n-1}{2} T_i \tag{4.2}$$

For the second part, with clock drift, the average latency to hear from the first CTS message from the receiver is $D_2 = T_i/2$. Summing up the two parts, $D = D_1 + D_2 = \frac{n}{2} \cdot T_i$.

Discovery latency of heterogenous case: Suppose two prime-grid quorum groups $\mathcal{Q}_{n\times n}$ and $\mathcal{Q}_{m\times m}$. Without loss of generality, let $n < m$.

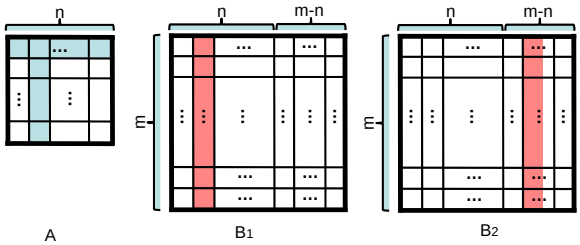


Figure 4.11: An example of the intersection between a big read quorum and a small write quorum. Case 1: intersection within the first n slots of A; case 2: intersection beyond the first n slots of A.

Theorem 17 (big write to small read) *Suppose $Q_j^r \subset \mathcal{Q}_{n\times n}$ and $Q_{j',1}^w \subset \mathcal{Q}_{m\times m}$ ($n < m$). The average discovery latency between two nodes that adopt Q_j^r and $Q_{j',1}^w$ for their LPL schedules, respectively, is $\frac{n}{2} \cdot T_i$.*

The proof of Theorem 17 is default to that for Theorem 16.

Theorem 18 (small write to big read) *Suppose $Q_{j',1}^w \in \mathcal{Q}_{n\times n}$ and $Q_j^r \in \mathcal{Q}_{m\times m}$ ($n < m$). The average discovery latency between two nodes that adopt Q_j^r and $Q_{j',1}^w$, respectively, is:*

$$D = \frac{n(m-n+1)}{2} \cdot T_i \tag{4.3}$$

Proof 12 *Similar to the proof of Theorem 16, the latency can be separated into two parts. We first derive the value of the latency part for reaching the first intersection.*

As shown in Figure 4.11, suppose the time difference between two nodes is Δ. Then $0 \leq \Delta < nT_i$ due to the rotation property of the grid quorum group. We consider two cases to derive this latency part:

Case 1: $1 \leq (j + \frac{\Delta}{T_i}) \mod n \leq n$ for $Q_j^r \subset \mathcal{Q}_{m\times m}$, i.e., the quorum B_1 in Figure 4.3. Suppose Q_j^r intersects with $Q_{j',1}^w$ in the k^{th} ($1 \leq k \leq n$) slot of $Q_{j',1}^w$. The corresponding latency is $(k-1) \cdot T_i$ and the probability is $1/m$. Thus, the average latency for this case is:

$$D_{case1} = \frac{1}{m}\sum_{k=1}^{n}(k-1)T_i = \frac{n(n-1)}{2m} \cdot T_i$$

Case 2: $n < (j + \frac{\Delta}{T_i}) \mod n \leq m$ for $Q_j^r \subset \mathcal{Q}_{m\times m}$, i.e., the quorum B_2 in Figure 4.3. For this case, Q_j^r will not overlap with $Q_{j',1}^w$ within the 1^{st} row of $\mathcal{Q}_{n\times n}$.

The intersection will happen in one slot of the read quorum $Q_{j'}^r \subset \mathcal{Q}_{n\times n}$, except for the first element of $Q_{j'}^r$.

The probability of $Q_j^r \subset \mathcal{Q}_{m\times m}$ overlapping with $(Q_{j'}^r)^p \subset \mathcal{Q}_{n\times n}$ (except for its first element and $p = \lceil m/n \rceil$) in the k^{th} ($1 < k \leq m$) slot of $(Q_{j'}^r)^p$ is $1/(m-1)$. The corresponding latency is $[n(k-1)+j^{'}] \cdot T_i$.

Thus, for each j that satisfies $n < (j + \frac{\Delta}{T_i}) \mod n \leq m$, the average latency of reaching the first intersection is: $\frac{1}{m-1}\sum_{k=2}^{m}[n(k-1)+j^{'}] \cdot T_i = (\frac{mn}{2} + j^{'}) \cdot T_i$.

The probability of Q_j^r to be in the j^{th} column satisfying $n < (j + \frac{\Delta}{T_i}) \mod n \leq m$ is $1/m$. Thus, this latency part can be expressed as:

$$D_{case2} = \frac{1}{m}\sum_{n}^{m}(\frac{1}{n}\sum_{j'=1}^{n}\frac{mn}{2} + j^{'})T_i \tag{4.4}$$

$$= [\frac{n(m-n)}{2} + \frac{(m-n)(n-1)}{2m}]T_i \tag{4.5}$$

Summing up the latency in case 1 and case 2, and considering the latency ($T_i/2$) to hear from the receiver within the intersected slot (which results from the unaligned boundary of time slots), we obtain, $D = D_{case1} + D_{case2} + \frac{T_i}{2} = \frac{n(m-n+1)}{2} \cdot T_i$.

Theorems 17 and 18 indicate that the average discovery latency between two nodes that adopt quorums from two prime-grid quorum groups is bounded.

4.2.4 Experimental Results

We implemented p-Grid in a WSN platform comprised of Telosb motes [10] running TinyOS 2.0, and experimentally evaluated the protocol performance. We implemented the protocol on top of the low power listening (LPL) interface in TinyOS 2.0. The radio Chipcon used by TelosB is CC2420 [10], which is complying with IEEE 802.15.4 standard.

We choose two competitors for comparison: X-MAC [27] with ALPL mode that is referred to as XMAC-ALPL in this section, and Disco [63]. We did not compare with B-MAC, since X-MAC is an improved version of B-MAC and it is fair to make comparison between XMAC-ALPL and p-Grid.

In our experiment, we used p-Grid for neighbor discovery in establishment of the shortest path routing protocol. We chose a small-sized network with 15 nodes among which there was one sink node. We randomly generated five topologies, and each data point presented in this section is the average of five topologies with ten runs on each topology. The radio range was configured to 10 meters (-25 dBm) according to the Telosb Datasheet [10].

The values of the dual preamble sampling periods (T_{P_1} and T_{P_2}) were set to $3\ ms$, which is compliant with the measurement in [13]. The duration of the preamble containing the DISC message (T_{DISC}) was set to 13 ms. The duration of the preamble containing the REPLY message

(T_{REPLY}) was set to 7 ms. Besides, we set other parameters in our experimental study as follows: DISC/REPLY message size was set to 128 bytes, and data packet size was 512 bytes.

Three important metrics were measured in our experimental study: (1) energy consumption ratio; and; (2) neighbor discovery latency; and (3) neighbor discovery ratio.

4.2.4.1 Energy Consumption Ratio

We first demonstrated the energy efficiency of p-Grid. For the purpose of easy measurement, we did not measure the actual energy consumed directly, but the average awake time for all nodes. In our experiment, a node went to listening mode immediately after completing neighbor discovery.

We set the length of one time slot as 50ms in p-Grid. We adopted the prime-grid quorum group $\mathcal{Q}_{3\times3}$ for read and write quorums, which has an energy consumption ratio of $\frac{1}{3}\cdot\frac{2*3}{50}=4\%$ in the idle state. To make a fair comparison, we also ameliorated XMAC-ALPL with dual-preamble mechanism by setting the length of one slot as 150ms (energy consumption ratio= $\frac{6}{150}=4\%$ in idle state). For the sake of heterogenous configuration, we chose the prime-grid quorum group $\mathcal{Q}_{5\times5}$ for some nodes, which has an energy consumption ratio of $\frac{1}{5}\cdot\frac{2*3}{50}=2.4\%$ in the idle state. For the XMAC-ALPL parameter setting of these nodes, we set the length of time slot as 250ms (energy consumption ratio= $\frac{6}{250}=2.4\%$ in idle state).

For the Disco protocol, we used the $(37,43)$ balanced primes and symmetric pairs ($\frac{1}{23}+\frac{1}{43}=6.6\%$ for cluster heads and cluster members, so that the total duty cycle was almost same as that in p-Grid and X-MAC. The slot length was set to 10ms.

Regarding the Disco protocol, we selected the $(23,43)$ balanced primes and symmetric pairs ($\frac{1}{23}+\frac{1}{43}=6.6\%$) for heterogenous nodes, respectively. Thus, the energy consumption ratio in idle state was almost the same as that in p-Grid and XMAC-ALPL. The slot length of Disco was set to 10ms.

We varied the neighbor discovery frequency from once per second to 5 times per second, we first measured the average awake time over reliable environment. As shown in Figure 5, the average awake time at probing node side increased with larger neighbor discovery frequency in p-Grid and XMAC-ALPL. The average awake time of Disco increased slightly since it was mainly determined by the prime selection without depending on specific neighbor discovery frequency. It is also shown that p-Grid consumed almost the same energy (i.e., awake time) as that for XMAC-ALPL, which coincides with our analysis in Section 4.2.3. Note that the awake time of p-Grid or XMAC-ALPL is higher than that of Disco in Figure 4.12. However, Disco achieves lower energy consumption at the cost of longer neighbor discovery latency and lower discovery ratio which will be shown later.

As to the performance comparison for unreliable environment, we set the node failure probability as 0.3 for all nodes in all mechanisms. As shown in Figure 4.12, at the probing node side, all nodes with p-Grid mechanism always took less awake time for discovering neighbors comparing with that in XMAC-ALPL, since all nodes with p-Grid took less time for sending preamble in order

to find out the receiver in case of link failures. At the neighbor sides, the awake time of p-Grid, XMAC-ALPL and Disco were almost identical.

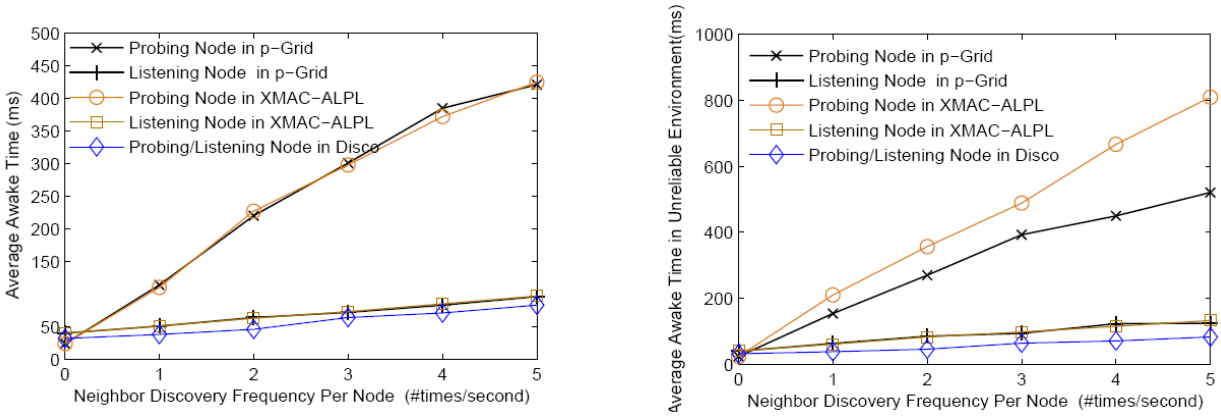


Figure 4.12: Energy Consumptions: reliable environment (left); unreliable environment with node failure probability being 0.3(right)

4.2.4.2 Neighbor Discovery Latency

We define neighbor discovery latency as the time delay between sending out DISC messages and receiving the REPLY message from a neighbor. The purpose of the evaluation in this Section was to verify the bounded neighbor discovery latency of p-Grid as claimed in Theorems 17and 18.

The protocol configurations for p-Grid, XMAC-ALPL, and Disco were the same as that in Section 4.2.4.1 to ensure that they had almost the same energy consumption ratio in the idle listening mode.

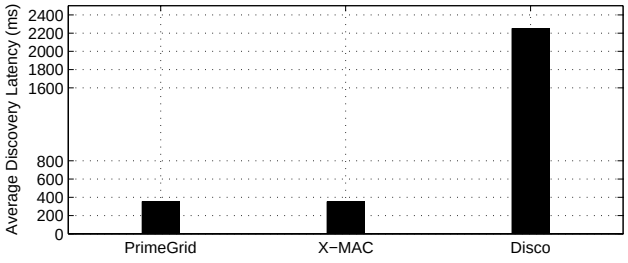


Figure 4.13: One-hop discovery latency between a read quorum and write quorum from the same grid quorum group $\mathcal{Q}_{5\times5}$

In Figure 4.13, we observe that p-Grid achieves slightly low latency than X-MAC. Although p-Grid may need more time slots for a node to intersect with its neighbor's schedule, p-Grid uses shorter time slots (e.g., 50ms), which can still guarantee lower discovery latencies. For Disco,

its discovery latency is significantly high, as it takes more than hundreds of slots for neighbor discovery.

In Figure 4.13, we observe that p-Grid achieved same neighbor discovery latency as XMAC-ALPL when they had same idle energy consumption ratio. According to our analysis in Section V-C, although p-Grid need more time slots for the schedule of a probing node to intersect with its neighbors schedule, p-Grid uses time slots with shorter duration (e.g., 50ms), which can still guarantee same discovery latencies as that of XMAC-ALPL. As to Disco, its discovery latency was the highest, as it took more than hundreds of slots for neighbor discovery which coincided with the simulation results in [63].

We also measured the neighbor discovery latency for asymmetric grid quorum groups. In this set of experiments, we varied the length of time slot from 50ms to 300 ms for different quorum groups. Figure 4.14 shows the impact of heterogeneity of quorum group on discovery latency. We observe that the discovery latencies were bounded with m^2 time slots for two nodes adopting write quorums from $\mathcal{Q}_{m\times m}$ and read quorums from $\mathcal{Q}_{n\times n}$ as their probing and listening schedules, which validated the claims in Theorems 17 and 18.

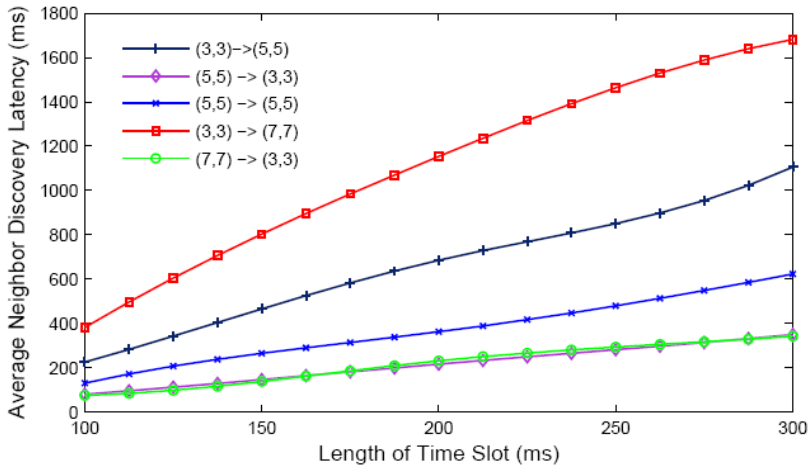
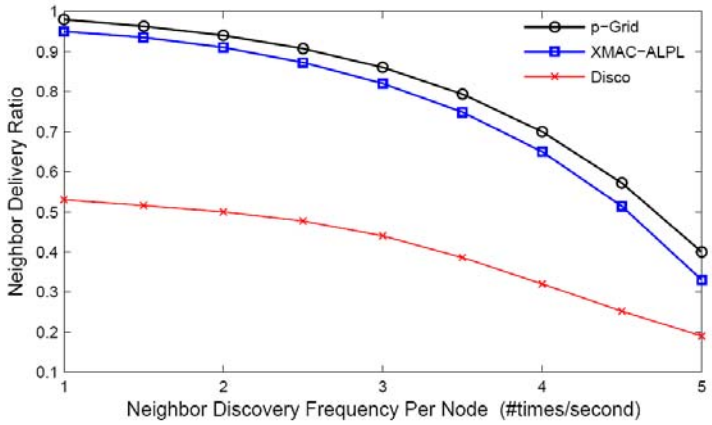


Figure 4.14: Average neighbor discovery latency for asymmetric design: $(m,m) -> (n,n)$ represents the discovery latency between a node equipped with write quorum from $\mathcal{Q}_{m\times m}$ and a node equipped with read quorum from $\mathcal{Q}_{n\times n}$

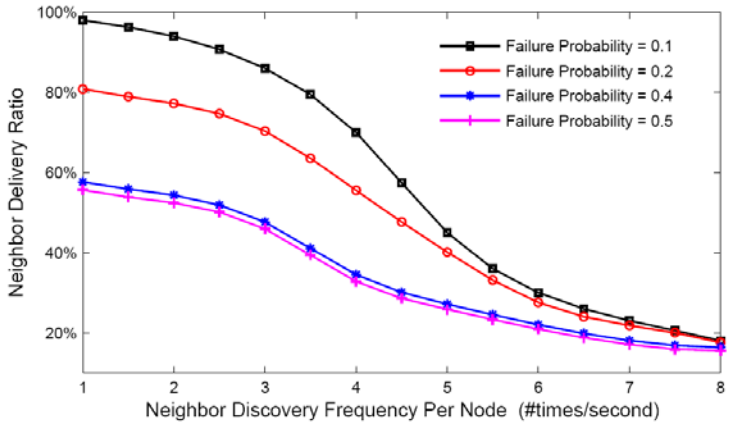
We also measured the end-to-end transmission delay under the chain topology which contained 6 nodes. We generated one packet every second at one end of the chain, and transmitted the packet over multi-hops to different destinations. We adopted quorums from the grid quorum group $\mathcal{Q}_{5\times 5}$ for all nodes in their LPL scheduling.

4.2.4.3 Neighbor Discovery Ratio

We define neighbor discovery ratio as the times successfully get a REPLY message in one cycle divided by the total times of neighbor probing by a node. We varied the neighbor discovery frequency from once per second to at most eight times per second.



(a) node failure probability = 0.1



(b) varying failure probability for p-Grid

Figure 4.15: Neighbor discovery ratio: a). different protocols; b). different failure ratios

We first fixed the node failure probability with 0.1 for all nodes and compared the performances. From Figure 4.15(a), we observe that p-Grid and XMAC-ALPL achieved higher delivery ratio than DISCO since both of them use the acknowledgement mechanism - i.e., the probing node sending the preamble and the neighbor acknowledging the receipt of the preamble (DISC message). In contrast, Disco achieved a lower delivery ratio because the long discovery latency of Disco resulted in the dropping of packets in the local memory buffer, especially for high neighbor discovery frequency

We also varied the node failure probability from 0.1 to 0.5 in order to evaluate the reliability of

p-Grid. In Figure 4.15(b), it is shown that the neighbor discovery ratio decreased with bigger node failure probability and that the ratio decreased with higher neighbor discovery frequency, which indicated that p-Grid cannot work well in highly unreliable environment.

4.2.5 Conclusions and Discussions

We presented a flexible solution, p-Grid, for neighbor discovery service in low duty-cycled WSNs. The motivation was to provide an energy-efficient, run-time configurable, asynchronous neighbor discovery mechanism for WSNs over unreliable environment. p-Grid is easy to implement: in idle state, nodes select a read quorum from a prime-grid quorum group as their low power listening schedules. For neighbor discovery, a node switches from listening mode to probing mode by selecting a write quorum from a prime-grid quorum group as its probing schedules, and sends out DISC messages in the time slots selected as write quorum. Armed with the prime-grid quorum group design, we show that there must be at least one overlapped slot in which both the probing node and its neighbors in listening mode are awake simultaneously.

It is possible that in p-Grid two or more REPLY messages collide at the probing node side, resulting in neighbor discovery failure. To address the collision, two strategies can be applied. One is that each node shifts its schedule by random time slots 8 after a time period so that nodes will not listen at the same time moment. Another one is by mutual neighbor discovery. A neighbor node also tries to detect the previous probing node and hence informs its presence to the previous probing node.

The design of p-Grid belongs to asynchronous neighbor discovery mechanisms so that time-synchronization is not necessary. Thus, it has the advantages of scalability. p-Grid can be integrated into existing MAC protocols or routing protocols to provide neighbor discovery service. We can also apply p- Grid for providing fault-tolerance functionality in WSNs. With some extensions, p-Grid can be adapted as a MAC protocol for delay-tolerant data communications in scenario where realtime data delivery is not mandatory.

In two-tiered WSNs, to meet heterogenous idle energy saving requirements, the cluster head should use a prime-grid quorum group with smaller size, and the cluster members should adopt a prime-grid quorum group with bigger size, because the discovery latency between quorums with smaller quorum size is shorter as shown in Section V. Thus, with a smaller grid quorum group for cluster heads, which may act as virtual routers in WSNs, the end-to-end transmission latency will be reduced resulting from shorter neighbor discovery latency. To compensate for imbalances in energy consumption for different nodes, the rotation mechanism of [94] can be applied by rotating the roles of all nodes.

Thus, p-Grid achieves performance trade-off between energy saving and neighbor discovery latency, which indicates lower energy consumption ratio with higher latency, and higher energy consumption ration with lower latency.

We envision several future directions for this work. One direction is to consider other quorum

systems, e.g., cyclic quorum systems, for the purpose of further reducing the discovery latency. Another direction is to combine the p-Grid protocol with a routing protocol to make an optimal cross-layer design for end-to-end data transmission.

Chapter 5

Routing over Heterogenous Wakeup Scheduling

5.1 Motivations

Multihop data routing over WSNs has attracted extensive attention in the recent years. Since there is no fixed infrastructure in sensor networks, the routing problem is different from the one in traditional wired networks or the Internet. Some routing protocols [91, 95] over WSNs presented in the literature are extended from the related approaches over wired/wireless ad-hoc networks. They usually find a path with the minimum hop count to the destination, which is based on the assumption that the link cost (or one-hop transmission latency) is relatively static for all wired/wireless links. However, for duty-cycled WSNs [13, 15, 27], that assumption may not be valid any more.

Duty-cycled WSNs include sleep-wakeup mechanisms, which can violate the assumption of static link costs. Currently, many MAC protocols support WSNs operating with low duty cycle, e.g., B-MAC [13], X-MAC [27]. In such protocols, sensor nodes operate in low power listening (or LPL) mode. In the LPL mode, a node periodically switches between the active and sleep states. The period for one active/sleep switching is called the LPL checking interval, which can be identical, or can be adaptively varied for different nodes, referred to as ALPL [18]. The duty-cycled mechanism have been shown to achieve excellent idle energy savings, scalability, and easiness in implementation. However, they suffer from time-varying neighbor discovery latencies, which is also pointed out by Ye *et.al.* [39]. As shown in Figure 5.1, the neighbor discovery latency between two neighbors is varying with different departure times. Even with synchronized duty-cycling, the neighbor discovery latency is varying at different time moments due to adaptive duty cycle setting as shown in Figure 5.1.

To formally define the problem, we first define the *link cost* as the time delay between data dispatching time at a sender and the data arrival time at the receiver. The link cost is time-varying in adaptively duty-cycled WSNs due to varying neighbor discovery latency, even though the physical

propagation condition does not change with time. The dispatching time means the time moment when the data is ready for transmission at the sender side. Thus, this raises a non-trivial problem: with time-varying link costs, how to find optimal paths with least nodes-to-sink latency for all nodes at all discrete dispatching time moments?

A similar problem has been modeled by previous works as the time-dependent shortest path problem (or TDSP) [50, 73] in the field of traffic networks [49], time-dependent graphs [74], and GPS navigation [75]. The general time-dependent shortest path problem is at least NP-Hard since it may be used to solve a variety of NP-Hard optimization problems such as the knapsack problem. However, depending on how one defines the problem, it may not be in NP, since its output is not polynomially bounded. Moreover, there are even continuous-time instances in which shortest paths consist of an infinite sequence of arcs as shown by Orda and Rom [96]. In the disseration, we study a special case where the networks are known as FIFO networks, in which commodities travel along links in a First-In-First-Out manner. Under the FIFO condition, the time-dependent shortest path problem is solvable in polynomial time.

The problem has also been studied with a distributed approach. For distributed solutions of the problem, the only previous work [50] computes the shortest paths for a specific departure time in each execution. If the whole time period has M discrete intervals (M is ∞ for infinite time intervals), we have to execute the algorithm in [50] M times, which is inefficient in terms of message complexity and time complexity, given the limited power and radio resource in WSNs. Therefore, the first motivation of our work is to design a fast distributed algorithm for the problem, which can efficiently enumerate all optimal paths with least end-to-sink latency for infinite time intervals.

The second motivation of our work is to propose an algorithm which can dynamically and distributively maintain time-dependent least-latency paths. In WSNs, a node may update its duty-cycle configuration (e.g., based on its residual energy), or join or leave the network, thereby changing the network topology. In such situations, the duty-cycle updating node or the joining/leaving node may change the cost of all the links with its neighbors, which means that a single node update can cause multiple link updates. Previous efforts on this problem [76, 77] are efficient in handling single link updates. Applying such solutions for multiple link updates would imply that multiple distributed updates execute concurrently for a single node update, which is not efficient in terms of message and memory space complexities for resource-limited WSNs.

The third motivation of our work is to address practical implementation issues. One is to understand how to address the awareness of duty cycling of neighborhood. We discuss several underlying mechanisms, such as B-MAC, S-MAC, and quorum-based wakeup scheduling, and propose a method for schedule awareness over them. Another is to understand how to simplify the vector presentation so that only smaller vector sizes are required, given the limited memory resource for embedded sensor nodes. We present a sub-optimal implementation, which achieves a trade-off between latency and memory usage. Finally, we discuss the complexities of our algorithms in some special scenarios, like static link costs and multiple sink nodes.

In this part of research, we first propose a distributed algorithm to compute the time-dependent

paths with least-latency for all nodes in a duty-cycled WSN. The algorithm has low message and space complexities. The algorithm is based on the observation that the time-varying link cost function is periodic, and hence by derivation, the time-varying distance function for each node is also periodic. We show that the link cost function satisfies the FIFO property [73]. Therefore, the time-dependent shortest path problem is *not* a NP-hard problem, and therefore is solvable in polynomial time. We also propose distributed algorithms for maintaining the shortest paths. The proposed algorithms re-compute the routing paths based on previous path information.

The message complexity of our algorithms is $O(\delta^2)$ per node update, where δ is the number of nodes that change either the distance or the parents in their shortest paths to the sink as a consequence of the corresponding nodes' update. The algorithms' space complexity is $O(maxdeg)$, which is scalable for large-scaled networks. Finally, we propose a sub-optimal implementation which requires vectors with smaller sizes to represent link cost functions and the distance function.

The contributions of our works on this parts are as follows: 1. We model adaptively duty-cycled WSNs as time-dependent networks. We show that such networks satisfy the FIFO condition and the triangular path condition; 2. We present distributed algorithms for finding the time-expanded shortest paths to the sink node for all nodes. When compared to the previous solution [50], our algorithms find the shortest paths in a single execution for infinite time intervals; 3. We present distributed shortest path maintenance algorithms with low message complexity and space efficiency; and 4. We propose sub-optimal implementation with vector compression.

5.2 Problem Formulations

We model an adaptively duty-cycled WSN as a directed graph $G = (V, E, C)$, with $|V|$ nodes and $|E|$ links. $C = \{\tau_{i,j}(t)|(i,j) \in E\}$ is a set of time-dependent link delays, i.e., $\tau_{i,j}(t)$ is a strictly positive function of time defined for $[0, \infty)$, describing the delay of a message over link (i, j) at time t. Each node n_i only knows the identity of the nodes in its neighbor set, defined as N_i.

We assume that time axes is arranged as consecutive short time slots. We denote the duration of one time slot for node n_i as T_i. It is possible that $T_i \neq T_j$ (ALPL) for two nodes n_i and n_j. The time expansion of each node n_i is modeled as discrete and infinite, where $\mathfrak{T}_i = \{t_i^0, t_i^1, t_i^2, \cdots, t_i^M\}$, M is $+\infty$, and $t_i^k - t_i^{k-1} = T_i$. We use the terms of checking interval and time slot interchangeably.

The wakeup schedule depends the underlying MAC protocols. We first assume a node can be operated with LPL mode where a node wakes up at the beginning of a time slot to check the channel state. If there is no activity, the node goes back to sleep, otherwise, it should stay awake. Then we relax the assumption and discuss how our works can be applied to other wakeup schedules like quorum schedules [1].

We consider the policy of no waiting at each node since the node-to-sink delay will not benefit from waiting. Thus, once the data arrives at an intermediate node, the node will try to dispatch the data immediately. Dispatching times are not the same as the data departure times, as the data may

still be buffered in the sender's memory. For simplicity in modeling and design, a node dispatches the received data at $t_i^k \in \mathfrak{T}_i$.

A nonnegative travel time $\tau_{i,j}(t_i^k)$ is associated with each link (i, j) with the following meaning: If t_i^k is the data dispatching time from node n_i along the link (i, j), then $t_i^k + \tau_{i,j}(t_i^k)$ is the data arrival time at node n_j.

The general problem of determining the shortest paths with the least latency in time-dependent WSNs can be defined as follows: Find the least time paths from all nodes to the sink node n_s corresponding to the minimum achievable delay d_i, $\forall n_i \in V$ and $\forall t_i^k \in \mathfrak{T}_i$, where

$$d_i(t_i^k) = \min_{n_j \in N_i} \{\tau_{i,j}(t_i^k) + d_j[t_i^k + \tau_{i,j}(t_i^k)]\} \tag{5.1}$$

Equation 5.1 is an extension of Bellman's equations [90] for the time-dependent network and is referred to as TD-Bellman's equation hereafter.

We assume that time slots in each node are numbered, and assume that the length of the data packets is fixed or relatively small, which is valid for a WSN designed for a specific application.

We also assume that a message arrives correctly with finite time from a sender to a receiver, which can be achieved by any reliable MAC-layer transmission mechanism. The network is not assumed to be time-synchronized, and nodes do not synchronize their wakeup/sleep schedules. However, we assume the awareness of wakeup schedules of neighborhood. We further assume that all nodes have the same time frequency, and their clocks drift at relatively slow speeds.

We do not explicitly consider radio interference in the problem modeling. We consume the interference can be concealed by sleeping, RTS/CTS mechanism, and multiple radio or frequencies.

5.3 Modeling Adaptively Duty-cycled WSNs

In this section, we model adaptively duty-cycled WSNs as time-dependent networks. We show that the link cost function is periodic and establish that the time-varying distance function is also periodic. Having done so, we then implement TD-Bellman's Equation by vector operations.

5.3.1 Link Cost Function

Without loss of generality, suppose there are two adjacent nodes n_i and n_j, where n_i is the sender and n_j is the receiver.

Suppose at time t_i^0, the neighbor discovery latency is $\Delta_{i,j}^{t_i^0}$. Then at time $t_i^k = t_i^0 + k * T_i$, the neighbor discovery latency can be expressed as:

$$\Delta_{i,j}^{t_i^k} = T_j - (k * T_i - \Delta_{i,j}^{t_i^0}) mod\, T_j \tag{5.2}$$

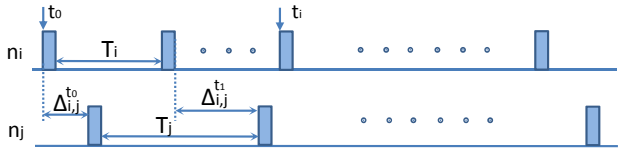


Figure 5.1: Varying neighbor discovery latency in heterogenous LPL mode

In an actual deployment, we can measure $\Delta_{i,j}^{t_i^0}$ in the following way: at the beginning of the time slot which starts at t_i^0, node n_i sends out a preamble which contains the node ID of n_j, and n_j immediately feeds-back an ACK containing the value of T_j once it receives the preamble. After receiving the ACK by n_i, the one-hop round trip delay from t_i^0 to the time at which the ACK is received is set to $\Delta_{i,j}^{t_i^0}$. Once we measured $\Delta_{i,j}^{t_i^0}$, $\Delta_{i,j}^{t_i^k}$ ($k \geq 0$) can be computed by Equation 5.2.

For data transmission with fixed length data packets, we define the data propagation time as τ_{data}. Now, for a directed link (i, j), we can set the link cost function as, for $\forall t_i^k \in \mathfrak{T}_i$,

$$\tau_{i,j}(t_i^k) = T_j - (k * T_i - \Delta_{i,j}^{t_i^0}) mod\ T_j + \tau_{data} \tag{5.3}$$

If τ_{data} is relatively small when compared with T_i and T_j, we can set $\tau_{data} = 0$. This is especially true for some WSN applications with small information reports, such as target tracking and environment monitoring.

Theorem 19 *For every link (i, j), the time-varying link cost function is periodic. The minimum period for the function regarding k is,*

$$P(\tau_{i,j}) = \frac{LCM(T_i, T_j)}{T_i} \tag{5.4}$$

where $\tau_{i,j}(t_i^k)$=$\tau_{i,j}(t_i^{k+P(\tau_{i,j})})$ ($k \geq 0$) and LCM is the least common multiple.

5.3.2 Distance to Sink

We refer to the node-to-sink delay as *distance*, for compatible representation with that in the static Bellman-Ford algorithm [90].

Consider node n_i and its neighbor n_j, where $T_i \neq T_j$. For dispatching time t_i^0 at n_i, let us suppose that the data arriving time at n_j is $t_j^{j_0}$. Then, for the dispatching time t_i^k at n_i, with the same time frequency for the two nodes, the corresponding time instant at n_j is $t_j^{j_0} - \Delta_{i,j}^{t_i^0} + k * T_i$. Hence, based on the neighbor discovery mechanism (i.e.,B-MAC [13] and X-MAC [27]), n_j will be discovered by n_i at the time instant $t_j^{j_0} + \lceil \frac{k*T_i - \Delta_{i,j}^{t_i^0}}{T_j} \rceil * T_j$, with respect to n_j's time clock. Therefore, the

function of distance of n_i to n_s from the path through n_j is:

$$d_i(t_i^k) = \tau_{i,j}(t_i^k) + d_j(t_j^{k^{'}+j_0}) \tag{5.5}$$

for $\forall t_i^k \in \mathfrak{T}_i$, where $k^{'} = \lceil (k * T_i - \Delta_{i,j}^{t_i^0})/T_j \rceil$ for $t_j^{k^{'}} \in \mathfrak{T}_j$, and j_0 is the data arriving time slot at n_j with respect to dispatching time t_i^0.

Theorem 20 *For a path $n_i \rightarrow n_{i-1} \cdots \rightarrow n_1 \rightarrow n_s$, the distance function $d_i(t_i^k)$,$\forall t_i^k \in \mathfrak{T}_i$, is a periodic function, where the minimum period for the function regarding k is:*

$$P(d_i) = \frac{LCM(T_0, T_1, \cdots, T_i)}{T_i} \tag{5.6}$$

for $d_i(t_i^k) = d_i(t_i^{k+P(d_i)})$, where $T_0, T_1, \cdots, T_i$ are durations of the LPL checking intervals of nodes $n_s, n_1, \cdots, n_i$, respectively.

Proof 13 *The proof is by induction. For $i = 1$, $d_1(t_1^k) = \tau_{1,s}(t_1^k) + d_s(t_1^{k^{'}})$. Since $d_s \equiv 0$, $d_1(t) = \tau_{1,s}(t_1^k)$. According to Theorem 19, $d_1(t_1^k)$ is periodic and its period is $LCM(T_0, T_1)/T_1$. Assume the claim is true for node n_{i-1}.*

Now for node n_i, $d_i(t_i^k) = \tau_{i,i-1}(t_i^k) + d_{i-1}(t_{i-1}^{k'+j_0})$, where $k^{'} = \lceil (k * T_i - \Delta_{i,j}^{t_i^0})/T_j \rceil$ and j_0 is the data arrival time slot at n_{i-1} with respect to t_i^k, based on Equation 5.5. Let us define $f_1 = \tau_{i,i-1}(t_i^k)$ and $f_2 = d_{i-1}(t_{i-1}^{k'+j_0})$. The minimum period for function f_1 is $P(f_1) = LCM(T_i, T_{i-1})/T_i$. Let $\prod = LCM(T_0, T_1, \cdots, T_{i-1})$. Based on the induction step, for distance function $d_{i-1}(t_{i-1}^k)$, the period is $\prod /T_{i-1}$. The period for function f_2 is $P(f_2) = \prod /gcd(\prod, T_i)$. Because $gcd(\prod, T_i) = \frac{\prod * T_i}{LCM(\prod, T_i)}$, we have $P(f_2) = \frac{LCM(\prod, T_i)}{T_i}$.

Therefore, the minimum period for $d_i(t_i^k)$ is:

$$P(d_i) = LCM[P(f_1), P(f_2)] = LCM(T_0, T_1, \cdots, T_i)/T_i$$

Given a WSN with different LPL checking intervals, the period for the distance function of any node is bounded by $LCM(T_0, T_1, \cdots, T_n)/\min\{T_i\}$, from Equation 5.6.

In practical implementations, it is recommended that $LCM(T_0, T_1, \cdots, T_n)/\min\{T_i\}$ is not arbitrarily large. Thus, our mechanism for finding the shortest paths at the routing layer should be based on cross-layer design. For example, $\{100\text{ms}, 200\text{ms}, 500\text{ms}, 1000\text{ms}\}$ is a good configuration set, where there is a bounded period $LCM(100, 200, 500, 1000)/min\{100, 200, 500, 1000\} = 10$ for the distance function. It means that for any node, its distance to the sink will repeat at most every 10 checking intervals.

5.3.3 Implementation via Vectors

We implement the discrete, periodic, and infinite link cost functions and the distance functions as vectors, and implement the TD-Bellman's equation 5.2 by vector operations.

We implement the link cost function $\tau_{i,j}(t_i^k)$ ($t_i^k \in \mathfrak{T}_i$) with a vector $\vec{\tau_{i,j}}$, where $|\vec{\tau_{i,j}}| = LCM(T_i, T_j)/T_i$ and $\tau_{i,j}[k]$ represents a set of numbers as follows:

$$\tau_{i,j}[k] = \{\tau_{i,j}(t_i^k), \tau_{i,j}(t_i^{k+|\vec{\tau_{i,j}}|}), \tau_{i,j}(t_i^{k+2*|\vec{\tau_{i,j}}|}), \cdots\} \tag{5.7}$$

For the node n_i, its distance function $d_i(t^k)$ ($t_i^k \in \mathfrak{T}_i$) can be implemented by $\vec{d_i}$, where $|\vec{d_i}| = P(d_i(t^k))$. $d_i[k]$ represents a set of numbers as follows:

$$d_i[k] = \{d_i(t_i^k), d_i(t_i^{k+|\vec{d_i}|}), d_i(t_i^{k+2*|\vec{d_i}|}), \cdots\} \tag{5.8}$$

However, there are two difficulties for the implementation of the TD-Bellman's equation by vector operations.

The first one is vector mapping. To implement Equation 5.5, even if we know $\vec{\tau_{i,j}}$ and $\vec{d_j}$, we cannot add up the two vectors directly. We define a new vector $\vec{d_j'}$ by

$$d_j'[k] = d_j[(k' + j_0)\ mod\ |\vec{d_j'}|] \tag{5.9}$$

where $k' = \lceil (k*T_i - \Delta_{i,j}^{t_i^0})/T_j \rceil$ and j_0 is the corresponding time slot for $\tau_{i,j}[0]$ at n_j (i.e., $t_i^0 + \Delta_{i,j}^{t_i^0} = t_j^{j_0}$).

Only after mapping $d_j[k]$ to $d_j'[k]$, we can add $\tau_{i,j}[k]$ to $d_j'[k]$. By vector mapping, the size for the new vector $\vec{d_j'}$ is:

$$|\vec{d_j'}| = \frac{|\vec{d_j}| * T_j}{gcd(|\vec{d_j}| * T_j, T_i)} \tag{5.10}$$

The second difficulty comes from the various sizes of vectors for link cost and distance. Suppose $d_i(\vec{j})$ is the vector representing the distance of n_i from a path through n_j in discrete time intervals. To implement Equation 5.5, if $\vec{\tau_{i,j}}$ and $\vec{d_j'}$ have the same size, we can directly add them up for computing $d_i(\vec{j})$. Otherwise, we need to expand the two vectors to be of the same size, which means expanding $\vec{\tau_{i,j}}$ by $LCM(|\vec{\tau_{i,j}}|, |\vec{d_j'}|)/|\vec{\tau_{i,j}}|$ times and $\vec{d_j'}$ by $LCM(|\vec{\tau_{i,j}}|, |\vec{d_j'}|)/|\vec{d_j'}|$ times. After the expansion, we can directly add up the expanded vectors. We call such an operation as *vector expansion*.

Vector mapping and expansion do not change the value of the discrete functions $\tau_{i,j}$ and d_j. They just change the representation of values of the two discrete functions. The vector expansion is valid since the time expansion is infinite.

We define the following functions for implementation:

Definition 18 *For a vector $\vec{v}$:*

- $ror(\vec{v}, ofs)$*: output* $\vec{v}'$ *where* $\forall k \in [0..|\vec{v}|-1]$, $v'[k] = v[(k+ofs)\ mod\ |\vec{v}|]$;
- $rol(\vec{v}, ofs)$*: output* $\vec{v}'$ *where* $\forall k \in [0..|\vec{v}|-1]$, $v'[k] = v[(\vec{v}|+k-ofs)\ mod\ |\vec{v}|]$;
- $map(\vec{v}, a, b, \Delta, ofs)$*: output* v' *where* $\forall k \in [0..|\vec{v}|-1]$, $v'[k] = v[(\lceil \frac{a*k-\Delta}{b} \rceil + ofs)\ mod\ |\vec{v}|]$;
- $exp(\vec{v}, e) = \vec{v}||\vec{v}...||\vec{v}\ (e\ times)$ *(|| presents catenating operation)*

We utilize these functions to implement the TD-Bellman's equation. Suppose that n_i has received the distance vector $\vec{d_j}$ of node n_j. Suppose $\tau_{i,j}[0]$ is associated with the time slot $l^0_{i,j}$ at node n_i and the data arriving time slot for $\tau_{i,j}[0]$ is l^0_j at n_j. Then, $d_i\vec{(j)}$, the distance vector of n_i to the sink from the path through n_j, can be calculated as:

$$\begin{aligned} \vec{d_j}' &= map[ror(\vec{d_j}, l^0_j), T_i, T_j, \tau_{i,j}[0], 0]; \\ d_i\vec{(j)}' &= exp(\vec{\tau_{i,j}}, e_1) + exp(\vec{d_j}', e_2); \\ d_i\vec{(j)} &= rol[d_i\vec{(j)}', l^0_{i,j}] \\ &= vec\_add(\vec{\tau_{i,j}}, \vec{d_j}, l^0_{i,j}, l^0_j) \end{aligned} \tag{5.11}$$

where $|\vec{d_j}'|$ is defined in Equation 5.10 and by defining $A = LCM(|\vec{\tau_{i,j}}|, |\vec{d_j'}|)$, $e_1 = A/|\vec{\tau_{i,j}}|$ and $e_2 = A/|\vec{d_j'}|$.

We update $\vec{d_i}$ and the corresponding parent vector $\vec{p_i}$ in the following way. Suppose the original $d_i[0]$ is associated with the time slot 0 at n_i (i.e., $d_i[0] = d_i(t^0_i)$). Now,

$$(\vec{d_i}, \vec{p_i}) = vmin\{exp(d_i, e_1'), exp(d_i\vec{(j)}, , e_2']\} \tag{5.12}$$

where $B = LCM(|d_i\vec{(j)}|, |\vec{d_i}|)$, $e_1' = B/|\vec{d_i}|$, and $e_2' = B/|d_i\vec{(j)}|$. The function $(\vec{d_i}, \vec{p_i}) = vmin(\vec{v_1}, \vec{v_2})$ compares the corresponding elements in the two vectors $\vec{v_1}$ and $\vec{v_2}$, and copies the smaller element of each pair into the corresponding element in $\vec{d_i}$ and the corresponding vector ID into $\vec{p_i}$.

In addition, we define an operator $\vec{<}$ for comparing two vectors $\vec{v_1}$ and $\vec{v_2}$. Let $C = LCM(|\vec{v_1}|, |\vec{v_2}|)$. If $\forall k \in [0..C-1]\ exp(\vec{v_1}, C/|\vec{v_1}|)[k] \leq exp(\vec{v_2}, C/|\vec{v_2}|)[k]$, then $\vec{v_1}\vec{<}\vec{v_2}$. For example, $[1,3]\vec{<}[2,3]$.

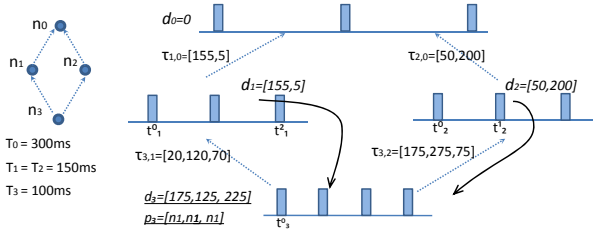


Figure 5.2: Example for vector implementation

Now, we give an example to illustrate the vector implementation. In Figure 5.2, node n_1 sends its distance vector d_1 to n_3 by a message containing d_1, and node n_2 sends its distance vector d_2 to n_3

by a message containing d_2, where $d_1 = [155, 5]$ and $d_2 = [200, 50]$. Suppose $\tau_{1,0}$, $\tau_{2,0}$,$\tau_{3,1}$ and $\tau_{3,2}$ are measured in their 0^{th} slot respectively.

After n_3 receives the message from n_1, it computes the distance vector $\vec{d_3(1)}$ based on Equation 5.11. We have $\vec{d_1'} = [155, 5, 155]$ and $\vec{d_3(1)} = [20, 120, 70] + [155, 5, 155] = [175, 125, 225]$. A similar computation will happen when receiving the message (d_2,1) and we can obtain $\vec{d_3(2)} = [375, 325, 275]$. Then, $vmin\{[175, 125, 225], [375, 325, 275]\}$ will return $([175, 125, 225], [n_1\, n_1\, n_1])$ containing the shortest distance and the corresponding parents for n_3.

5.3.4 Properties

Theorem 21 FIFO condition*: The link cost function $\tau_{i,j}(t_i^k)$ satisfies the FIFO property, which means, for any $t_i^{k_1} < t_i^{k_2}$,*

$$t_i^{k_1} + \tau_{i,j}(t_i^{k_1}) \leq t_i^{k_2} + \tau_{i,j}(t_i^{k_2}) \tag{5.13}$$

Proof 14 *From Equation 5.3, we have $t_i^{k_2} + \tau_{i,j}(t_i^{k_2}) - t_i^{k_1} - \tau_{i,j}(t_i^{k_1}) = (k_2 - k_1) * T_i + (k_1 * T_i - \Delta_{i,j}^{t_i^0})\ mod\ T_j - (k_2 * T_i - \Delta_{i,j}^{t_i^0})\ mod\ T_j = (k_2 - k_1) * T_i + [(k_1 - k_2) * T_i]\ mod\ T_j = (k_2 - k_1) * T_i - [(k_2 - k_1) * T_i]\ mod\ T_j \geq 0$.*

By Theorem 21, the time-dependent shortest path problem in asynchronous duty-cycled WSNs is not NP-hard and is solvable in polynomial-time [73].

Theorem 22 *Suppose node n_i has two neighbor n_j and n_k which are one-hop away from each other. Then, at a time instant, t_i, we have the triangular property:*

$$\tau_{i,j}(t_i) \leq \tau_{i,k}(t_i) + \tau_{k,j}[t_i + \tau_{i,k}(t_i)] \tag{5.14}$$

Proof 15 *Suppose at time t_i, the data arriving time slot at n_j is t_j, and the data arriving time at n_k is t_k.*

If $t_k \leq t_j$, which means that the data arriving time at n_k is earlier than the data arriving time at n_j, $\tau_{i,k}(t_i) + \tau_{k,j}[t_i + \tau_{i,k}(t_i)] = t_k - t_i + t_j - t_k = t_j - t_i = \tau_{i,j}(t_i)$.

If $t_k > t_j$, which means that the data arriving time at n_k is later than the data arriving time at n_j, we have $\tau_{i,k}(t_i) + \tau_{k,j}[t_i + \tau_{i,k}(t_i)] = t_k - t_i + t_j^{'} - t_k > t_j - t_i + t_j^{'} - t_k > t_j - t_i > \tau_{i,j}(t_i)$. The theorem follows.

Theorem 28, as illustrated in Figure 5.3(a), illustrates that node n_i will always arrive at its neighbor n_j directly without through other nodes. We have the following claim.

Lemma 5 Triangular Path Condition*: For a node n_i and its neighbor n_j, at any dispatching time, the one-hop path $n_i \rightarrow n_j$ always has the least time delay for data transmission.*

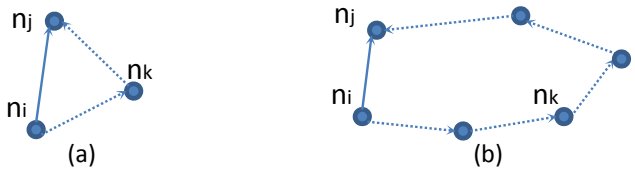


Figure 5.3: Triangular path condition: the direct one-hop path $n_i \rightarrow n_j$ always achieve the least latency among all pathes from n_i to n_j.

Proof 16 *We prove this by induction. Suppose there are multiple nodes along the path from n_i to n_j and define these nodes as $n_k, n_{k+1}, \cdots, n_{k+i}$. If $i = 0$, based on Theorem 28, Lemma 6 is true.*

Now assume $i = k$ and Lemma 6 is true. We prove that Lemma 6 is true when $i = k + 1$. Suppose the adjacent node to n_i along the path is n_k. Then the direct path $n_k \rightarrow n_j$ has shorter latency comparing the path along $n_{k+1}, \cdots, n_{k+i}$. Also based on Theorem 28, $n_i \rightarrow n_j$ has shorter latency then the path $n_i - > n_k - > n_j$. Therefore, the direct path $n_i \rightarrow n_j$ has shorter latency than the path along $n_k, n_{k+1}, \cdots, n_{k+i}$. The lemma follows.

An illustration is given in Figure 5.3(b). Note that the triangular path condition does not exist in static networks.

5.4 Algorithm for Initial Route Construction

We now present distributed algorithms for initial time-dependent shortest path route construction in duty-cycled WSNs, where the distances from all nodes to the sink node are initially infinite.

The proposed algorithm referred to as the FTSP algorithm, Fast Time-Dependent Shortest Path algorithm, is inspired by existing algorithms [50], and is adapted from the distributed Bellman-Ford algorithm augmented with β-synchronizer [97]. Comparing with the solution in [50], which only computes the shortest paths for a given specified discrete time, FTSP compute the shortest paths for infinite discrete time intervals in one execution.

Let $|D_m|$ denote the diameter of the longest shortest path for all nodes. We show that the message complexity of FTSP is $O(|D_m||E|)$ and the time complexity is $O(|D_m||V|)$. FTSP does not suffer from exponential message complexity, like in previous work [98] for the static shortest path problem over asynchronous networks.

5.4.1 Distributed Algorithm Description

We present the data structures and message formats in FTSP:

- $\vec{d_i}$: vector of distance from n_i to n_s, defined in Equation 5.8; initially all elements are ∞;

- $\vec{\tau_{i,j}}$: link delay from n_i to its neighbor n_j, defined in Equation 5.7; $\tau_{i,j}[0]$ is obtained by measurement;
- $\vec{p_i}$: vector of parents for n_i in the shortest path $n_i \rightarrow n_s$ for infinite time intervals; initially all elements are node n_i;
- $MSG(ID_{src}, ID_{from}, \vec{d}, updated)$: control message; ID_{src} is the node ID of sink node, or update source node (see Section 5.5); ID_{from} is the sender's node ID; $\vec{d}$ is the distance vector of the sender; $updated$ is to show whether there is update in the current iteration, which will be explained later;
- $ACK[j]$: boolean indicating whether a node receives a control message from its neighbor n_j

We assume that n_i knows the duration of the checking interval T_j of all its neighbors $n_j \in N_i$ after measuring link delays. Initially, a directed spanning tree rooted at n_s is built upon the network. We assume that n_i knows its parent st_p_i in the spanning tree. We also assume that FTSP is invoked by higher-level protocols that create "START" impetuses at n_s.

Algorithm 1: Algorithm for sink node n_s in FTSP:

Initialization:
 $\forall n_j \in N_s$, $ACK[j] = 0$, $updated[j] = true$;
On receiving START:
 send MSG(s, s, 0, false) to $\forall n_j \in N_s$;
On receiving MSG(j,$\vec{d_j}$, l_j, bChanged):
 $ACK[j] = 1$; $updated[j] = bChanged$;
 if *(*$\forall n_j \in N_s, ACK[j] == 1$*)* **then**
  $ACK[j] = 0$;
  if *(*$\forall n_j \in N_s$, $updated[j] == false$*)* **then**
   STOP;
  else
   send MSG(s, s, 0, false) to $\forall n_j \in N_s$;

The first iteration of FTSP begins when node n_s receives the "START" impetus. Subsequent ones begin whenever n_s completes an iteration and determines whether another one is necessary by checking whether there is a node whose distance was minimized in the last iteration.

Each iteration begins at node n_s by sending a control message to all its neighbors. When replies from all its neighbors have been received, node n_s concludes that an iteration is completed. Every other node, i.e., n_i ($n_i \neq n_s$), begins an iteration upon receiving a control message from its parent st_p_i in the spanning tree, upon which it sends control messages to all its neighbors except st_p_i. When replies are received from all these neighbors, a control message is sent to the parent, thereby completing the current iteration at node n_i.

The control message from node n_j contains the distance vector $\vec{d_j}$ (known thus far during the previous iterations) between n_j and n_s. When such a message is received at node n_i, node n_i checks whether the new information minish the value of any element in the current distance vector. It does so by considering the path that goes through n_j, taking into account the most recent information from n_j.

We describe the procedures in Algorithms 1 and 2.

Algorithm 2: Algorithm for node $n_i (i \neq s)$ in FTSP:

Initialization:
$st\_p_i = NULL$;
for $\forall n_j \in N_i$ **do**
    $ACK[j] = 0$; $updated[j] = true$;
    Measure link delay $\Delta^0_{i,j}$ at the beginning of any time slot;
    $l^0_{i,j}$ = the time slot number for measurement;
    l_j = the data arriving time slot number at n_j;
    for $\forall k \in [0..LCM(T_i, T_j)/T_j]$ **do**
        $\tau_{i,j}[k] = T_j - (k * T_i - \Delta^0_{i,j}) mod\, T_j$;

On receiving MSG(s, j, $\vec{d_j}$, bChanged) from n_j:
$ACK[j] = 1$; updated[j] = bChanged; $\vec{d_i}^{prev} = \vec{d_i}$;
if $n_j == st\_p_i$ **then**
    send MSG(s, i, $\vec{d_i}$, false) to $\forall n_j \in N_i$ except st_p_i;
$d_i\vec{(j)}$ = vec_add($\vec{d_i}, \vec{d_j}, l^0_{i,j}, l_j$); /* Equation 5.11*/
$(\vec{d_i}, \vec{p_i})$ = vmin($\vec{d_i}, d_i\vec{(j)}$); /* Equation 5.12*/
if ($\vec{d_i} \vec{<} \vec{d_i^{prev}}$) **then** updated[j] = true;
if ($\forall n_j \in N_i, ACK[j] == 1$) **then**
    if ($\exists n_j \in N_i, updated[j] == true$) **then** bUpdated = true;
    else then bUpdated = false;
    send MSG(s, i, $\vec{d_i}$, bUpdated) to st_p_i;
    $\forall n_j \in N_i, ACK[j] = 0$;

The functions $vec\_add(\cdot)$, $vmin(\cdot)$, and operator $\vec{<}$ are defined in Section 5.3.3.

5.4.2 Correctness and Complexity

Let $PATH(i, t_i^k)$ be a path obtained by node n_i, which is starting at time t_i^k and moving along its parent $p_i[k]$. Let $d_i[k]_m$ denote the value of $d_i[k]$ after the m^{th} iteration in FTSP. We have the following properties:

Theorem 23 *1. After termination, $PATH(i, t_i^k)$ is loop-free and concatenated; and 2. In the m^{th} ($m \geq 0$) iteration, a node n_i whose shortest path is at most m-hop away from the sink node for all discrete time intervals $t_i^k \in \mathfrak{T}_i$ will be determined.*

Proof 17 *For part 1, $\forall t_i^k \in \mathfrak{T}_i$, after termination, suppose $p_i[k]$ is set to the node n_j for the shortest path with respect to n_s. Since n_j is the parent of n_i at the time slot t_i^k, $PATH(i, t_i^k)$ is a path composed of $PATH(j, t_i^k + \tau_{i,j}(t_i^k))$ which is appended to node n_i $\forall t_i^k \in \mathfrak{T}_i$. Thus, for any node n_i and $\forall t_i^k \in \mathfrak{T}_i$, $PATH(i, t_i^k)$ is concatenated.*

We prove that $PATH(i, t_i^k)$ is loop-free by contradiction. Without loss of generality, assume that there is a loop and the loop is $(n_{i_0} \rightarrow n_{i_1} \rightarrow n_{i_2} \cdots n_{i+k} \rightarrow n_{i_0})$. This means that there is a

shortest path $(n_{i_0} \rightarrow n_{i_1} \cdots, \rightarrow, n_{i+k})$, where n_{i+k} is one-hop away from n_{i_0}. According to the triangular path condition in Lemma 6, such a shortest path cannot exist because $n_{i_0} \rightarrow n_{i+k}$ is always the shortest one among all paths from n_{i_0} to n_{i+k}, contradicting the assumption.

We prove part 2 by induction on m. It is easy to find that the claim is true for $m = 0$. Now, assume that the claim is true for $m - 1$ (i.e., the inductive hypothesis). We prove for m by induction.

Consider a specific time t_i^k and a node n_i such that there is a shortest path with at most m hops between n_i and n_s. Let $SP(i, s, t_i^k, m)$ be the shortest path which is at most m hops from n_i to n_s at time t_i^k. Let n_j be n_i's parent on $SP(i, s, t_i^k, m)$ at t_i^k. This means that there is a path with at most $m - 1$ hops between n_j and n_s.

*By the inductive hypothesis, n_j determined its shortest distance at the $(m-1)^{th}$ iteration. In the m-*th *iteration, node n_j sends its minimized distance vector $\vec{d}_{j_{m-1}}$ to node n_i. Thus, $SP(i, s, t_i^k, m)$ is determined after receiving the vector $\vec{d}_{j_{m-1}}$ in the m^{th} iteration, which completes the inductive step. Since t_i^k is chosen arbitrarily, this holds for all values of $t_i^k \in \mathfrak{T}_i$.*

The part 2 in Theorem 23 implies that, for $m \geq D_m$, all nodes determine their minimum delay and the corresponding parents for all time intervals, since all shortest paths contain at most D_m nodes.

Theorem 24 *The message and time complexity of FTSP is $O(D_m|E|)$ and $O(D_m|V|)$ respectively.*

Proof 18 *Based on the implementation of the β-synchronizer in [99], in each iteration, there is exacted one message traversing each link in the spanning tree, totally $|E|$ messages exchanged. By Theorem 23, the number of iterations is upper bounded by the longest shortest path's length D_m. Thus, the message complexity is $O(D_m|E|)$.*

Suppose the largest delay for transmitting a message in the spanning tree is a constant, denoted by $|C|$. In each iteration, the time consumed is at most $|V| * |C|$. Since there are at most D_m iterations, the time complexity is $O(D_m|V|)$.

5.5 Algorithm for Dynamic Route Maintenance

When compared with static networks, link changes and node changes are more frequent in duty-cycled WSNs. If a node changes its duty-cycle configuration, or dynamically joins or leaves the network, the links connecting with all its neighbors will be changed at multiple time intervals. In such a situation, a single node update usually causes multiple link updates.

Some previous works (e.g., [77]) in static networks have proposed solutions that efficiently deal with single link updates. They are inefficient for multiple link updates caused by a single node update. The algorithms in [77] are also memory-inefficient, since each node stores the route entries for all other nodes, incurring the space complexity of $O(|V|)$.

The proposed algorithms, referred as to FTSP-M ("M" meaning maintenance), focus on per node update and can be easily extended to node insertion and deletion. If there are multiple node updates, the algorithms will run concurrently at multiple nodes.

5.5.1 Distributed Algorithm Descriptions

The proposed distributed algorithms for path maintenance are also equipped with $\beta-$synchronizer. We use similar data structures and message formats as that in Section 5.4.1.

Suppose the source update node is n_u and the corresponding input change is σ. We divide σ into two parts: σ_{inc} and σ_{dec}, where σ_{inc} includes the increasing links for $\forall t_u^k \in \mathfrak{T}_u$, and σ_{dec} includes the decreasing links for $\forall t_u^k \in \mathfrak{T}_u$. Let $\delta(\sigma_{inc})$ be the set of nodes that change either the distance or the parents for all infinite discrete time intervals, as a consequence of σ_{inc}. Similarly, let $\delta(\sigma_{dec})$ be the set of nodes affected by σ_{dec}. Apparently, $\delta(\sigma) = \delta(\sigma_{inc}) \cup \delta(\sigma_{dec})$.

FTSP-M consist of two phases for node n_u and all nodes $n_i \in \delta(\sigma)$ as described in Algorithms 3 and 5.

In phase 1, an initial spanning tree is built up gradually to contain all nodes in $\delta(\sigma_{inc})$. The purpose of phase 1 is to let all nodes in $\delta(\sigma_{inc})$ increase their distances to the sink node as a consequence of σ_{inc}, along the time-expanded shortest path trees rooted at n_u. After the termination of phase 1, all nodes in $\delta(\sigma)$ will never increase their distance again.

The rationale here is to satisfy the sufficient loop-free condition claimed in [100]: If at time t, node n_i detects a link-cost decrease or a decrease in the distance reported by a neighbor, then node n_i is free to choose its new parents. This is referred to as the *distance increase condition* (or DIC).

In each iteration, node n_u sends a control message to all neighbors. Every other node, i.e., n_i ($n_i \neq n_u$) will send control messages to all its neighbors if it is in the spanning tree and receives a message from its parent. If n_i is not in the spanning tree in the current iteration, it checks whether n_j is its parent in its shortest path after receiving a control message from n_j, which can be done by checking whether $n_j \in \vec{p_i}$. If true, n_v will join the spanning tree and set $newsp_i = n_j$. By doing so, the spanning tree will increase at most one level in each iteration.

A control messages will traverse from the root (n_u) to all other nodes in the spanning tree just like that in FTSP. When node n_i receives a control message from n_j, it only updates the element to be increased in its distance vector, as illustrated in Algorithm 4.

When replies from all its neighbors have been received, node n_u concludes that an iteration is completed. When replies are received from all neighbors by a node, a control message is sent to the parent, thereby completing the iteration at the node. When there is no distance increase for all nodes in $\delta(\sigma_{inc})$, phase 1 will be terminated and node n_u will start phase 2.

In phase 2, the initial spanning tree built up in *phase 1* is continuously growing until it contains all nodes in $\delta(\sigma)$. Phase 2 is also running by iterations. In each iteration, when a node n_i not in the spanning tree receives a control message from n_j, if the value of any elements in its distance vector can be minished, it will join the spanning tree by setting its parent $newsp_i$ to n_j. The distance update and message traversing in phase 2 of FTSP-M are just similar to that in FTSP.

Algorithm 3: Operations at update node n_u in FTSP-M:

Initialization:

Same initialization as in Algorithm 2;
if $p_u \neq null$ **then**
  for $\forall n_j \in N_u$ **do**
    ($\vec{d_j}$,l_j^0) = get_dist(n_j); /* retrieve $\vec{d_j}$, detailed implementation is omitted */
    $\vec{d_j}(u)$=vec_add($\vec{\tau_{u,j}}$,$\vec{d_j}$,$l_{u,j}^0$,l_j^0); /* Equation 5.11*/
    inc_update($\vec{d_u}$,$\vec{p_u}$, n_j, $\vec{d_j}(u)$);

Phase 1: On receiving START:

send MSG(u,u,$\vec{d_u}$,false) to $\forall n_j \in N_u$;

Phase 1: On receiving MSG(u, j, $\vec{d_j}$, bChanged):

$ACK[j] = 1$; $updated[j] = bChanged$;
if *(*$\forall n_j \in N_u, ACK[j] == 1$*)* **then**
  if *(*$\forall n_j \in N_u$, $updated[j] == false$*)* **then**
    Beginning Phase 2;
  else
    send MSG(u,u,$\vec{d_u}$,false) to $\forall n_j \in N_u$;
    $\forall n_j \in N_u, ACK[j] = 0$;

Phase 2: On Beginning Phase 2:

send MSG(u,u,$\vec{d_u}$,false) to $\forall n_j \in N_u$;

Phase 2: On receiving MSG(0, j, $\vec{d_j}$, bChanged) from n_j:

$ACK[j] = 1$; $updated[j] = bChanged$; $d_u^{\vec{prev}} = \vec{d_u}$;
$d_u\vec{(j)}$ = vec_add($\vec{d_u}, \vec{d_j}, l_{i,j}^0, l_j,$); /* Equation 5.11 */
($\vec{d_u}, \vec{p_u}$) = vmin($\vec{d_u}, d_i\vec{(j)}$); /* Equation 5.12 */
if ($\vec{d_u} \vec{<} d_u^{\vec{prev}}$) updated[j] = true;
if *(*$\forall n_j \in N_u, ACK[j] == 1$*)* **then**
  if *(*$\forall n_j \in N_u$, $updated[j] == false$*)* **then**
    STOP;
  else
    send MSG(u,u,$\vec{d_u}$,false) to $\forall n_j \in N_u$;
    $\forall n_j \in N_u, ACK[j] = 0$;

Algorithm 4: Function: inc_update($\vec{d_1}$,$\vec{p_1}$,n_2,$\vec{d_2}$)

d_1 = exp($\vec{d_1}$,($|\vec{d_1}| * |\vec{d_2}|$)/$|\vec{d_1}|$); d_2 = exp($\vec{d_2}$,($|\vec{d_1}| * |\vec{d_2}|$)/$|\vec{d_2}|$);
p_1 = exp($\vec{p_1}$,($|\vec{d_1}| * |\vec{d_2}|$)/$|\vec{d_1}|$); flag = false;
for $k = 0$ **to** $|\vec{d_1}| * |\vec{d_2}| - 1$ **do**
  if $p_1[k] == n_2$ && $d_1[k] < d_2[k]$ **then**
    $d_1[k] = d_2[k]$; flag = true;
return flag;

Algorithm 5: Operations in node $n_i (n_i \neq n_u)$ in FTSP-M:

Initialization: $newsp_i = null$; $\forall n_j \in N_i$,$updated[j] = false$;

Phase 1: On receiving MSG(u, j, $\vec{d_j}$, bChanged) from n_j:

if $n_j \in N_u$ **then**
  re-measure $\tau_{i,u}$ at the beginning of one time slot;
  reset $l^0_{i,j}$ and l^0_u;

$\vec{d_i}(j)$=vec_add($\vec{\tau_{i,j}}$,$\vec{d_j}$,$l^0_{i,j}$,l^0_j);
if $newsp_i == null$ **then**
  if (inc_update($\vec{d_i}$,$\vec{p_i}$,n_j,$\vec{d_i}(j)$)) **then**
    $newsp_i = n_j$; send MSG(u,i,$\vec{d_j}$,true) to n_j;
  else send MSG(u, i, $\vec{d_j}$, false) to n_j;
else
  FORWARD(u); /* u means the MACRO is executed in phase 1 */
  $ACK[j] = 1$; updated[j] = inc_update($\vec{d_i}$,$\vec{p_i}$,n_j,$\vec{d_i}(j)$);
  ACK_REPLY(u);

Phase 2: On receiving MSG(0, j, $\vec{d_j}$, bChanged):

if $newsp_i == null$ **then**
  if ($updated[j]$) **then**
    $newsp_i = n_j$; send MSG(0, i, $\vec{d_j}$, true) to n_j;
  else send MSG(0, i, $\vec{d_j}$, false) to n_j;
else
  FORWARD(0); /* 0 means the MACRO is executed in phase 2 */
  $ACK[j] = 1$; $updated[j] = bChanged$; $d_i^{\vec{prev}} = \vec{d_i}$;
  $\vec{d_u}(j)$ = vec_add($\vec{d_u}, \vec{d_j}, l^0_{i,j}, l_j$); /* Equation 5.11*/
  ($\vec{d_u}, \vec{p_u}$) = vmin($\vec{d_u}, \vec{d_i}(j)$); /* Equation 5.12*/
  if ($\vec{d_u} \vec{<} d_u^{\vec{prev}}$) updated[j] = true;
  ACK_REPLY(0);

FORWARD(int dict): code macro

if ($newsp_i == n_j$) **then**
  send MSG(dict,i,$\vec{d_i}$,false) to $\forall n_j \in N_i$ except $newsp_i$;

ACK_REPLY(int dict): code macro

if ($\forall n_j \in N_i, ACK[j] == 1$) **then**
  if ($\exists n_j \in N_i, updated[j] == true$) **then** bUpdated = true;
  else then bUpdated = false;
  send MSG(dict, i, $\vec{d_i}$,bUpdated) to $newsp_i$;
  $\forall n_j \in N_i, ACK[j] = 1$; bUpdated = false;

5.5.2 Correctness and Complexity

In phase 1, all nodes in $\delta(\sigma_{inc})$ do not change their parents, but increase their distances as a consequence of σ_{inc}. Thus, there is no loop in phase 1. In phase 2, all nodes in $\delta(\sigma)$ will never increase their distances, thereby satisfying the distance increase condition [100]. All paths are therefore loop-free.

Theorem 25 *In phase 1, each node in $\delta(\sigma_{inc})$ with at most m hops away from n_u along the time-dependent shortest path will not increase its distance after m iterations.*

Proof 19 *The proof is through induction on m. It is easy to find that the claim is true for $m = 0$. Now, assume that the claim is true for $m - 1$ (i.e., the inductive hypothesis). We prove for m by induction.*

Consider a specific time t_i^k and a node $n_i \in \delta(\sigma_{inc})$ such that there is a shortest path with at most $m - 1$ hops between n_i and n_s after node update. Let $SP(i, u, t_i^k, m - 1)$ be the shortest path which is at most $m - 1$ hops from n_i to n_u at time t_i^k. Let n_j be n_i's neighbor which is not updated yet due to σ_{inc}. Then n_j is at most m hops away from n_u.

*By the inductive hypothesis, n_i will not increase its distance after $(m - 1)^{th}$ iteration. In the m-*th *iteration, node n_i sends its updated distance vector $\vec{d}_{j_{m-1}}$ to node n_j, and n_j will determine its updated distance in the iteration if $n_j \in \delta(\sigma_{inc})$. Thus, $SP(j, u, t_i^k, m)$ is determined after receiving the vector $\vec{d}_{im-1}$ in the m^{th} iteration, which completes the inductive step. Since t_i^k is chosen arbitrarily, this holds for all values of $t_i^k \in \mathfrak{T}_i$.*

By Theorem 25, after $|\delta(\sigma_{inc})|$ iterations, all nodes in $\delta(\sigma_{inc})$ will not increase their distance anymore.

Definition 19 *Updated-subpath: for any node $n_i \in \delta$, the updated-subpath is from n_i to the first node n_e not in δ along the shortest path from n_i to n_u.*

Theorem 26 *In phase 2, all generated updated-subpaths are loop-free, and updated-subpaths with at most m hops long are determined in the m^{th} iteration.*

Proof 20 *After phase 1, the DIC loop-free condition [100] is satisfied. Thus, all updated-subpaths are loop free in phase 2.*

The proof for at most m hops-long updated-subpaths being determined in the m^{th} iteration can be done by induction. It is easy to find that the claim is true for $m = 0$. Now, assume that the claim is true for $m - 1$ (i.e., the inductive hypothesis). We prove for m by induction.

Consider a specific time t_i^k and a node $n_i \in \delta$ such that there is a updated-subpath which contains m hops between n_i and n_u. Let $UP(i, u, t_i^k, m)$ be the updated-subpath which is at most m hops long at time t_i^k. Let n_j be n_i's parent on $UP(i, u, t_i^k, m)$ at t_i^k. This means that there is a updated-subpath with at most $m - 1$ hops between n_j and n_s.

*By the inductive hypothesis, the updated-subpath from n_j to n_u is determined at the $(m-1)^{th}$ iteration. In the m-*th *iteration, node n_j sends its minimized distance vector $\vec{d}_{j_{m-1}}$ to node n_i. Thus, $UP(i,u,t_i^k,m)$ is determined after receiving the vector $\vec{d}_{j_{m-1}}$ in the m^{th} iteration, which completes the inductive step. Since t_i^k is chosen arbitrarily, this holds for all values of $t_i^k \in \mathfrak{T}_i$.*

Theorem 27 *The message complexity for per node update with $\delta(\sigma)$ output change is $O(|\delta(\sigma)|^2 * maxdeg)$. The time complexity is $O(|\delta(\sigma)|^2)$, and space complexity is $O(maxdeg)$.*

Proof 21 *In phase 1, the number of iterations is $\delta(\sigma_{inc}) \leq \delta(\sigma)$. In phase 2, there are at most $\delta(\sigma)$ iterations before all updated-subpaths are decided. Thus totally, there are $O(|\delta(\sigma)|^2 * maxdeg)$ messages. In each iteration, the consumed time is at most $|\delta(\sigma)| * C$ (C is the largest transmission delay for all links). Thus, the message complexity is $O(|\delta(\sigma)|^2)$. Since a node only stores the information of all its neighbors, the space complexity is $O(maxdeg)$.*

5.5.3 Sub-optimal Implementation with Vector Compression

The key implementation of our proposed algorithms is the vector representation of link cost functions and distance functions. However, if the vector size is too large (i.e., the LCM in Equation 5.6 is too large), the proposed algorithms, FTSP and FTSP-M, may not be feasible given the limited memory resource of embedded sensor nodes. For example, if the vector size is larger than the total number of nodes, one could instead exchange the information about wake-up period and offset directly. Then, the optimal forwarding decision can be computed at the source node when event occurs, using centralized dynamic programming. Such a solution may have even lower overhead when the vector size is large.

The largest length of the vector is determined by $\frac{LCM(T_0,T_1,\cdots,T_i)}{T_i}$, as shown in Equation 5.6. In a real implementation, to avoid arbitrary long vectors, there are two possible solutions: 1. Use a predefined duty cycle set, so that the $\frac{LCM(T_0,T_1,\cdots,T_i)}{T_i}$ can be bounded by carefully selecting a duty cycle set, as discussed in Section 5.3.1 and Section 5.3.2; 2. Adopt vector compression to achieve a trade-off, i.e., adopt a low accurate distance vector, which takes less memory space to represent the end-to-end latency. Hence, the output path is sub-optimal in terms of latency. For the second method, we may not need a bounded global $\frac{LCM(T_0,T_1,\cdots,T_i)}{T_i}$.

We propose to use vector compression when the size of the vectors are large. The basic idea is to smooth all values in a vector and represent the vector with less information. For example, for a vector $[1234567]$ with 7 elements, we can approximately represent the vectors by a vector with 2 elements, such as $[(\frac{1+2+3+4}{4},4),(\frac{5+6+7}{3},4)]$. Each tuple (v,s) in the vector represents the average value of s elements in the original vector.

The formal description of vector compression is as follows.

Vector compression: Suppose the source vector is $v_s = [v_1, v_2, \cdots, v_n]$ and the target vector size

is m ($n > m$). We compress v_s by:

$$v_t = [(\frac{v_1 + v_2 + \cdots + v_{len}}{len}, len), (\frac{v_{1+len} + v_{2+len} + \cdots + v_{2*len}}{len}, len), \cdots,$$
$$(\frac{v_{1+(m-1)*len} + v_{2+(m-1)*len} + \cdots + v_n}{n-(m-1)*len}, n-(m-1)*len)]$$

where $len = \lceil \frac{n}{m} \rceil$.

We choose the average value $\frac{v_{1+i*len}+v_{2+len}+\cdots+v_{(i+1)*len}}{len}$ as the value of $v_t[i]$ because the expected deviation can be minimized by,

$$P = \sum_{j=1}^{len} \frac{1}{len} |v_t[i] - v[j]|$$

However, other filters, such as the Wavelet transform filter [101] can also be applied for vector compression, as typically done in image compression.

5.5.4 Further Discussions

The FTSP algorithm described in Section 5.4 is a proactive routing protocol. Although its time complexity is $O(D_m * |V|)$ for initial route construction, it is affordable in the initial stage of WSNs. The low space complexity ($O(|maxdeg|)$) for route maintenance makes the algorithm scalable for large-scale WSNs.

Note that FTSP and FTSP-M target the ALPL mode [18] with various checking intervals. When all nodes have homogenous LPL checking intervals (like that in the standard B-MAC), according to Equations 5.3 and 5.5, the link cost function and the distance function will become constants. In such a case, our algorithms will default to the static shortest path algorithm. However, FTSP and FTSP-M will yield the same message complexities and time complexities for the static situation.

The dissertation focused on the scenario of single sink node. However, it can be extended to multiple destinations. In WSNs, there is usually no end-to-end communication between two arbitrary nodes. We only consider the generalization of communication between one node and multiple sink nodes rather than the communication between two arbitrary nodes. In such a generalization, a node has to store the route information for all destinations (i.e., sinks). For each destination, our proposed algorithms can be applied for the route construction. Note that the delay property is maintained for each individual destination in such an extension.

5.6 Experimental Results

We evaluated the performance of FTSP, FTSP-M, and the sub-optimal implementation through extensive simulations using the OMNET++ discrete event simulator [102]. We compared our al-

gorithms against other related algorithms for the TDSP problem, including the distributed Bellman-Ford algorithm [98] adapted to the time-dependent model (Section 5.3), referred to here as TD-Bellman, and DSPP1 [50]. The following three major metrics were measured in the evaluation: 1. message count; 2. time cost, which is the total time slots needed for stabilization; and 3. average memory cost.

We examined two main factors that affect the performance of our algorithms, including network size and underlying duty cycle setting. Our experimental settings were compatible with typical configurations, as in [18, 39, 91]. The wireless communication range in our simulation was set to 10m. We adopt the wireless loss model used in [103], which considers the oscillation of radio links. The size of the data packets was fixed as 512 bytes.

We generated 8 network size sets with varying sizes, G_1,$\cdots$,G_8, which are listed in Table 5.1. For each network size, we randomly generated 10 topologies. Each data point presented in our experimental results in this section is the average of 10 topologies, with 10 runs on each topology.

Table 5.1: Network Size Sets

	G_1	G_2	G_3	G_4
$\|V\|$	50	80	200	400
	G_5	G_6	G_7	G_8
$\|V\|$	600	1K	1.5K	2K

Table 5.2: Time slot sets

C_1 (ms)	$\{100, 100, 100, 100\}$
C_2 (ms)	$\{100, 200, 300, 600\}$
C_3 (ms)	$\{100, 200, 400, 800\}$
C_4 (ms)	$\{100, 200, 500, 1000\}$

We choose two MAC protocols: ALPL (adaptive low power listening) and quorum-based duty-cycling. The ALPL mode means that a node just wakes up for a short time during a checking interval to check the channel activities. The duration of the checking interval varies for different nodes. We changed the duration of the checking interval in our simulation experiments with 4 sets, C_1, C_2, C_3, and C_4, as listed in Table 5.2. With each set, we randomly choose one element as the value of the LPL checking interval for each node.

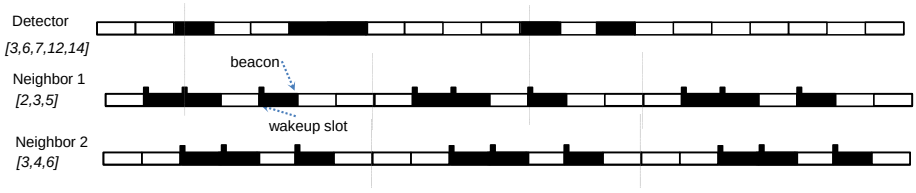


Figure 5.4: Quorum duty-cycling: detector with the quorum from (21,5,1)-difference set design, and neighbors with quorum from (7,3,1)-difference set design, as presented in [1].

For quorum-based duty-cycling, we choose the $(7, 3, 1)$, and $(21, 5, 1)$ difference sets for the heterogenous wakeup schedule settings. The duration of one time slot was set to 100 ms in quorum-based duty-cycling. Since FTSP, FTSP-M, TD-Bellman, and DSPP1 are independent of wakeup scheduling, we argue that the comparison is fair even when we choose quorum-based duty-cycling.

5.6.1 Least-latency Path Construction

In the first set of simulation experiments, we tested the ALPL mode and chose C_4 as the time slot setting, which indicates that the largest distance vector size is 10 by Equation 5.6, and varied the network size. With the number of nodes increasing from 50 to 2000 in G_1,$\cdots$,G_8, the average time consumed and the message count are shown in Figure 5.5.

The average execution time of DSPP1 is about 10 times larger than that of FTSP, since DSPP1 has to be executed 10 times to compute the shortest paths for all time intervals. FD-Bellman is better than FTSP when the network size is small, since FD-Bellman does not have a distributed synchronizer in its execution. When the network size becomes large (i.e., $\geq$ 1K), FTSP outperforms FD-Bellman due to the exponential worst case message complexity of Bellman-Ford algorithm. We observed similar trends for time cost for the three algorithms, as shown in Figure 5.5.

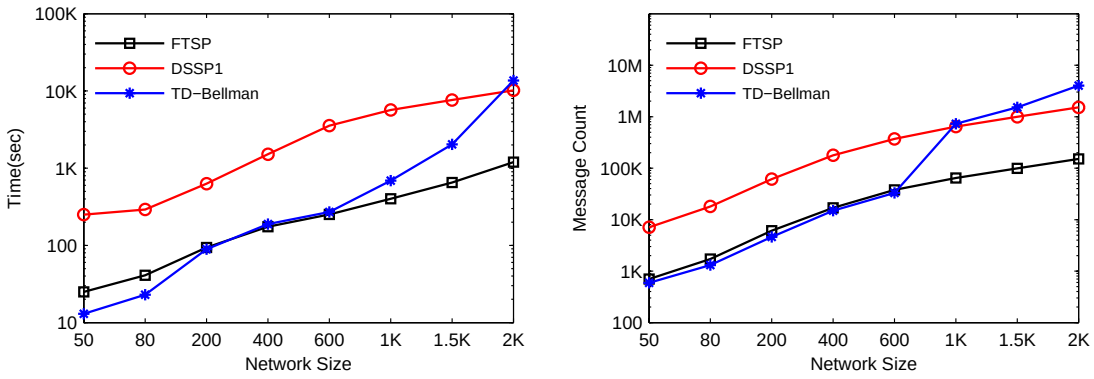


Figure 5.5: Comparison of time efficiency and message efficiency by varying $|V|$

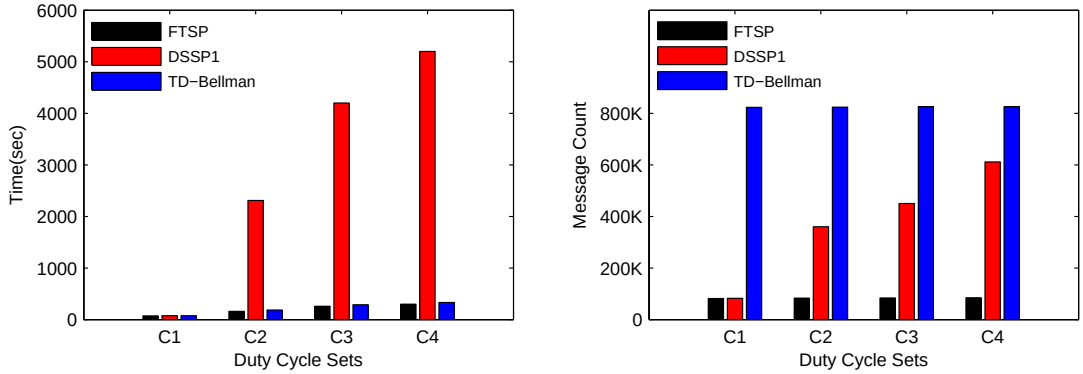


Figure 5.6: Comparison of time efficiency and message efficiency by varying time slots in ALPL mode

We also varied the time slot sets with a fixed network size of G_6, which represents medium-sized WSNs. The results are shown in Figure 5.6. We observe that FTSP and FD-Bellman do not change

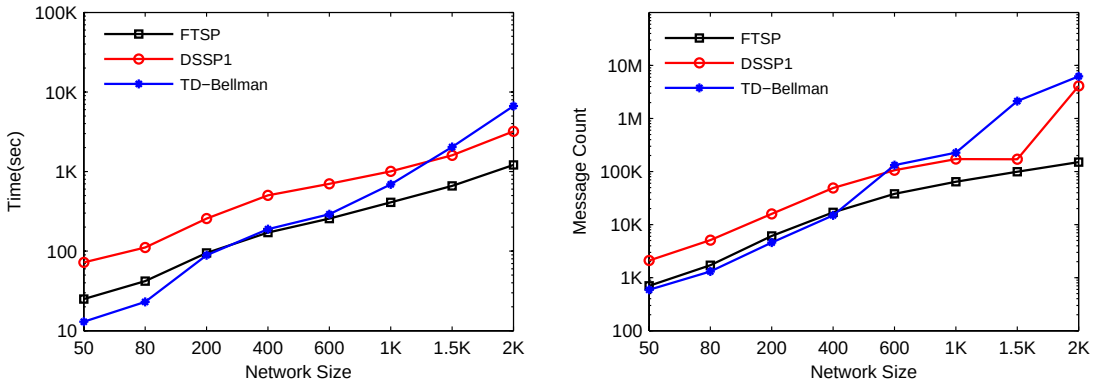


Figure 5.7: Comparison of time efficiency and message efficiency for quorum-based duty-cycle setting

their message count significantly since they only depend on the network size. The time cost of all algorithms become worse when the average value of all elements in the selected time slot set becomes larger, since the average link delay is correspondingly increasing.

Finally, we tested the performance for quorum-based duty-cycling by fixing the network size of G_6. Each node randomly chose the $(7, 3, 1)$ and $(21, 5, 1)$ difference sets for its heterogenous schedule settings. As shown in Figure 5.7, we observe similar trends for execution time and message count. The average execution time of DSPP1 is about 3 times larger than that of FTSP, since DSPP1 has to be executed 3 times to compute the shortest paths for all time intervals given the $(7, 3, 1)$ and $(21, 5, 1)$ difference sets. The time costs for all the algorithms become worse when the network size becomes larger, which is consistent with the conclusion in Theorem 24.

5.6.2 Least-latency Path Maintenance

For evaluating the path maintenance performance of FTSP-M, we return to static networks by selecting the time slot set of C_1 in Table 5.2 for the ALPL mode. We do so for the purpose of a fair comparison against previous works in [77], referred to here as Full-Dynamic and DSDV [104] with unicast support (multicast is not well-supported in duty-cycled WSNs).

We first evaluate the effect of input changes on all algorithms for medium-sized networks by choosing G_6. Figure 5.8 shows the results. We observe that FTSP-M achieves the median message cost and time cost when input changes become large. The reason is that DSDV may suffer exponential message complexity. In addition, FTSP-M uses the synchronizer which consumes additional time and messages, which is not as efficient as that in Full-Dynamic.

We also measure the average memory cost by varying the network size for the two underlying duty-cycling mechanisms. We first chose the ALPL mode and set the time slot set of C_1 in Table 5.2 for all nodes. The results shown in Figure 5.9 indicate that FTSP-M achieves the best memory cost, which does not depend on the network size. The memory costs of DSDV and Full-Dynamic

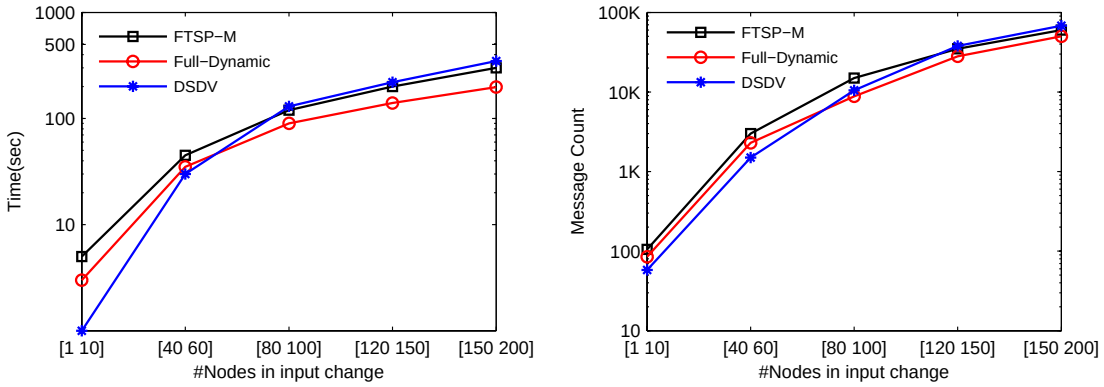


Figure 5.8: Performance comparison for route maintenance by varying input change: ALPL mode

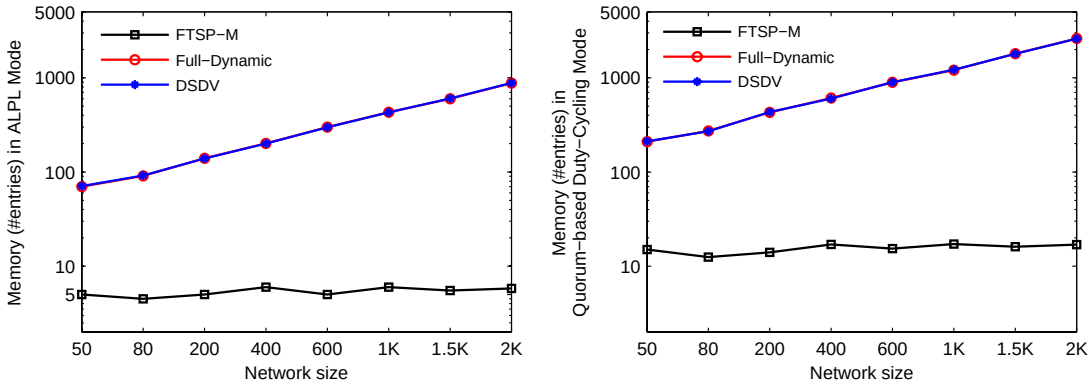


Figure 5.9: Performance comparison for route maintenance on memory required in each node

increase with the network size, since each node stores an entry for all other nodes.

Finally, we tested the average memory cost for the quorum-based duty-cycling mechanism. Each nodes randomly chose the $(7, 3, 1)$ difference sets for the heterogenous schedule settings. As shown in Figure 5.9, we observe similar trends for the quorum-based duty-cycling mechanism. The only difference is that with $(7, 3, 1)$ difference sets, each node maintains 3 entries in the routing table for all neighbors.

5.6.3 Performance of Sub-Optimal Implementation

For sub-optimal implementation with vector compression, the performance is a trade-off between path latency and message size. We evaluated this tradeoff for both initial route construction and route maintenance. We first fixed the duty cycle setting by choosing C_4 and compared the performance between FTSP and its sub-optimal implementation. Since the message count does not rely on vector implementation, we compared the vector size. Figure 5.10 shows the results. We observe that the sub-optimal implementation achieves less message size with vector compression. To

understand the end-to-end latency of the two techniques, we compared the maximum value of the least latency achieved by all nodes (defined as max-min latency). As shown in Figure 5.10, the sub-optimal implementation has higher max-min latency compared with FTSP. However, sub-optimal implementation outperforms FTSP on average message size.

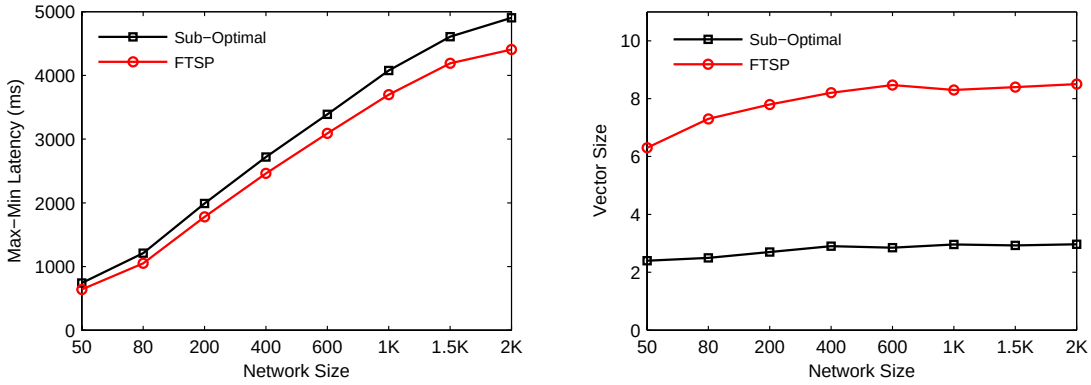


Figure 5.10: Performance comparison for route construction: latency and vector size over ALPL mode

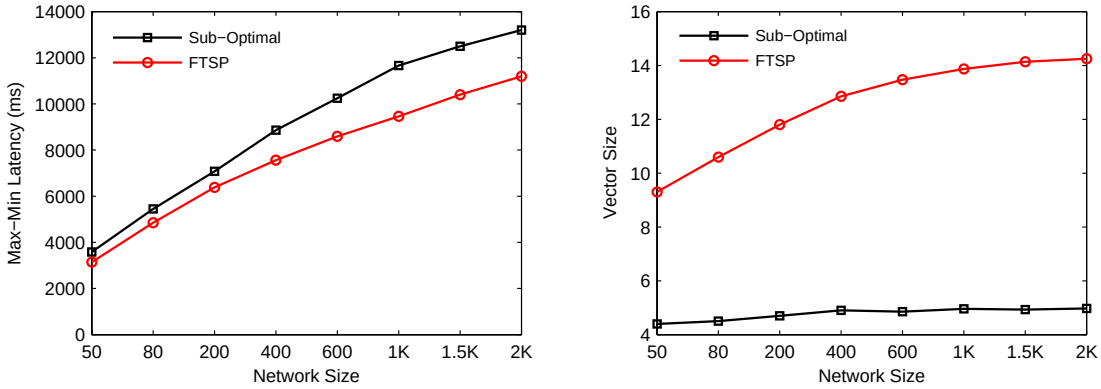


Figure 5.11: Performance comparison for route construction: latency and message size over quorum-based duty-cycling mode

We then fixed the duty cycle setting by choosing the $(7, 3, 1)$ and $(21, 5, 1)$ difference sets as the wakeup schedule for all nodes. We observed similar trends as shown in Figure 5.11. We observe that the sub-optimal implementation has higher max-min latency compared with FTSP, but has a smaller vector size.

5.7 Discussions

In this section, we discuss some practical implementation issues. In particular, we discuss how to achieve awareness of schedules of neighborhood nodes and how to reduce the vector size given a

large LCM in Equation 5.6.

FTSP and FTSP-M require awareness of wakeup schedules of the neighborhood. Achieving this requirement depends on the specific underlying MAC protocol. We discuss three scenarios for achieving schedule awareness over different MAC protocols.

Active Neighbor Discovery: This means that a node needs to probe the schedules of its neighbors actively. We consider two category of mechanisms, which needs active neighbor discovery. One is the LPL mode as adopted by B-MAC [13] and X-MAC [27]. The other is the low duty-cycling mode where time axes are arranged as consecutive short time slots, and all slots have the same duration T_s. In the low duty cycling mode, each node n_i adopts a periodic wakeup schedule every n_i time slots. The wakeup schedule can be once every n_i (n_i is an integer) slots. n_i is called the cycle length for node n_i. For example, in Figure 6.1(a), neighbor 1 has a schedule of $\{1, 0, 0\}$, where 1 means wakeup slot and neighbor 2 has the schedule of $\{1, 0, 0, 0\}$. We assume that beacon messages are send out at the beginning of wakeup slots, similar to [1, 85]. When a node wants to detect schedules of its neighborhood, it will wait until beacons are received from its neighbors.

The problem of configuring the schedule (i.e., the value of L_i) has been studied in the past (e.g., [15]), and is outside the scope of this dissertation.

Quorum-based Duty-Cycling: In addition, FTSP and FTSP-M can be applied for asynchronous quorum-based wakeup scheduling [12] in wireless sensor/ad-hoc networks. In asynchronous wakeup scheduling, the neighbor discovery delay may vary at different time moments as illustrated in Figure 5.12.

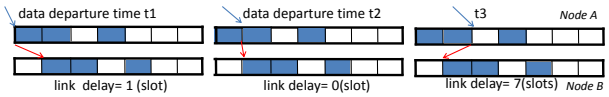


Figure 5.12: Varying neighbor discovery delay for quorum wakeup scheduling

The time-varying link cost function and the distance function are also periodic, and FTSP and FTSP-M are directly applicable.

Synchronization: MAC protocols with synchronization require that all neighboring nodes wake up at the same time. The simplest method for doing this is to use a fully synchronized pattern, like that in the S-MAC protocol [23]. In this case, all nodes in the network wakeup at the same time according to a periodic pattern. A further improvement can be achieved by allowing nodes to switch off their radio when no activity is detected for at least a timeout value, like that in the TMAC protocol [24]. In this scheme, neighboring nodes form virtual clusters to set up a common sleep schedule. The main disadvantages of such scheduled neighbor discovery schemes are the complexity of implementation and the overhead for synchronization. Through synchronization, a node can conveniently know the schedules of its neighbors. Schedule awareness can be achieved by periodic message exchange between a node and its neighbors.

5.8 Conclusions

In this chapter, we addressed the distributed shortest path routing problem in duty-cycled WSNs. Our contributions are four-fold. First, we modeled duty-cycled WSNs as time-dependent networks, which satisfy the FIFO condition. Second, we presented FTSP algorithm for finding shortest paths in such networks. FTSP has polynomial message complexity and is more time-efficient than previous solutions. Third, we presented FTSP-M for distributed route maintenance with node insertion, updating, and deletion. FTSP-M is memory efficient and has polynomial message complexity. Finally, we proposed a sub-optimal implementation in order to reduce the vector size. The vector size of the sub-optimal solution does not depend on the largest LCM value as shown in Equation 5.6. Experimental results validated the effectiveness and efficiency of our solutions.

We envision several directions for future work. One is to investigate the time-dependent minimum spanning tree problem, which is NP-Hard in duty-cycled WSNs. Another direction is to study time-dependent multicast routing in duty-cycled WSNs, which is a required service for many applications and is the reverse direction of all-to-one least-latency routing. And it is also valuable to consider interference in the future research.

Chapter 6

Broadcast over Duty-Cycled WSNs

6.1 Motivations and Goals

Multihop broadcast [40] is an important network service in WSNs, especially for applications such as code update, remote network configuration, route discovery, etc. Although the problem of broadcast has been well studied in always-on networks [41, 42] such as wireless ad hoc networks where neighbor connectivity is not a problem, broadcast is more difficult in duty-cycled WSNs where each node stays awake only for a fraction of time slots and neighborhood nodes are not simultaneously awake for receiving data. The problem becomes more difficult in asynchronous [12] and heterogenous duty-cycling [43] scenarios.

To support broadcast, synchronization of wakeup schedules is one promising approach adopted by many duty-cycling MAC protocols, such as S-MAC [23] and T-MAC [24]. Such protocols simplify broadcast communication by letting neighborhood nodes stay awake simultaneously. However, this approach results in high overhead for periodic clock synchronization when compared to the low frequency of broadcast service in WSNs. Since energy is critical to WSNs, energy-efficient asynchronous MAC protocols have become increasingly attractive for data communication, as proposed in B-MAC [13], RI-MAC [85], Disco [63], and quorum-based wakeup scheduling [1,12].

However, previous asynchronous MAC protocols for duty-cycled WSNs mostly focus on unicast communication, and do not work well for broadcasting. One straightforward way to support one-hop broadcast in such cases is to deliver data multiple times for all neighbors, which results in redundant transmissions. With multihop broadcasting to an entire network, the problems are more amplified, as some neighbors attempt to forward the broadcast message while the original transmitting node still attempts to transmit it to other nodes of its neighbors, increasing collisions and wasting energy consumption for transmission.

There have been some efforts in the past to support multihop broadcasting in duty-cycled WSNs. Wang *et al.* [83] transformed the problem into a shortest-path problem with the assumption of duty-cycle awareness, which is not valid for asynchronously duty-cycled WSNs. DIP [105], ADB [40], and opportunistic flooding [51] were designed with a smart gossiping approach. Essentially, these

protocols use unicast to replace broadcast for flooding, toward reducing the flooding latency in the entire network. However, they may lack efficiency in large-scale networks or on delivering large chunks of data to entire network because message cost and higher transmission energy consumption.

To overcome the disadvantages of replacement via pure unicast, we present Hybrid-cast, an asynchronous broadcast protocol for broadcasting with low latency and reduced message count. In Hybrid-cast, a node only forwards a message to neighbors who wake up and send out beacon messages. A node defers broadcasting by one or more time slot(s) after receiving the beacon message from the first awake neighbor in order to wait for more nodes that may potentially wake up, so that more nodes are accommodated in one broadcast. It also adopts online forwarder selection in order to reduce the transmission redundancy. Compared with previous protocols, Hybrid-cast can achieve less broadcast latency and smaller message count.

Goals: Let us define the broadcast latency as the time between the beginning of a broadcast and the time at which every node receives the broadcast message. Also, let us define the broadcast count as the number of broadcasting via all nodes to ensure that the entire network receives the message. Our goal is to design a broadcast schedule, which can not only shorten the broadcast latency but also the broadcast count for flooding a message to the entire network. The protocol that we present, Hybrid-cast, is a heuristic solution to this problem.

6.2 Hybrid-cast Protocol

6.2.1 Overview

In Hybrid-cast, a transmitter will stay awake for long enough time to hear the beacon message from its neighbors. Due to heterogenous wake-up scheduling, for low duty-cycling, the node will stay awake for L_m time slots, which is the largest cycle length of all neighbors. By doing this, it can hear beacons from all neighbors. For quorum duty-cycling, the transmitter will switch to the wakeup schedules which has the largest cycle length from all its neighbors.

Hybrid-cast adopts opportunistic forwarding with delivery deferring to shorten broadcast latency and broadcast count: the transmitter will forward the message within δ time after it hears the beacon messages from early-wakeup neighbors, rather than forwarding immediately after hearing the beacon messages. An illustration is given in Figure 6.1(a). Here, δ (i.e., $\delta = T_s$ for low duty-cycling) is called the deferring time. By deferring, the first-awake neighbor can still receive the broadcast message. Meanwhile, more neighbors which wake up during the deferred time period can receive the broadcast message, so that less number of broadcast is necessary for one-hop broadcasting.

To further reduce redundant transmissions, Hybrid-cast adopts online forwarder selection. “Online” means that a node selects the least relay node among its instant one-hop awake neighbors, rather than all one-hop neighbors, to cover its two hop neighbors, in order to reduce transmission

redundancy and collision.

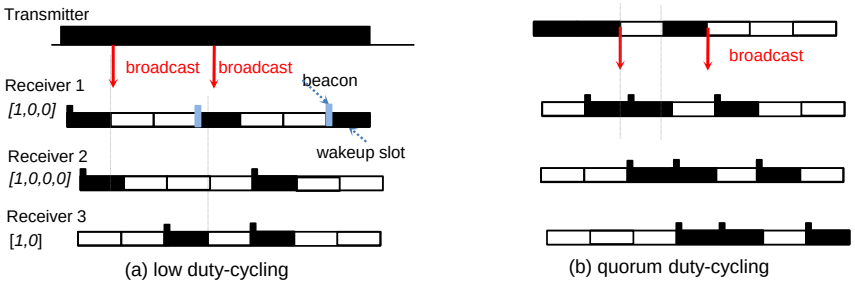


Figure 6.1: Opportunistic broadcasting with delivery deferring (a) low duty-cycling case; (b) quorum duty-cycling case with wakeup schedules of [1,1,0,1,0,0,0] and its rotations which comply with (7,3,1) cqs design in [1].

6.2.2 Wakeup Schedule Switching

Due to adaptive duty-cycling, neighbor discovery becomes more difficult. In order to hear the beacon message from all neighbors, a node must switch its wakeup schedule for staying awake for enough time slots.

For the case of low duty-cycling, in the idle state, a node n_i follows its own wakeup schedule. If the node needs to forward a broadcast message (i.e., the node is selected as a relay node), n_i should stay awake for at least L_m slots, where $L_m = \max_{n_j \in N(i)}\{L_j\}$. By doing this, n_i can hear beacon messages from all neighbors within the minimum necessary time slots.

For the case of quorum duty-cycling, n_i just switches to the schedule of the node which has the longest cycle length. Due to the non-empty intersection property [1], n_i can still hear all neighbors even when it does not stay awake in every time slot of a whole cycle length.

A node needs to know the largest cycle length of its neighbors before schedule switching. This can be achieved by either pre-setting the largest global cycle length or by dynamic neighbor information exchange protocols.

6.2.3 Opportunistic Forwarding with Deferring

Opportunistic forwarding means that a transmitter forwards data immediately to the neighbor which wake up earlier, for minimizing broadcast latency. Previous efforts on opportunistic flooding such as [51] use unicast for broadcasting. However, opportunistic forwarding via pure unicast suffers from large broadcast count.

In Hybrid-cast, broadcast deferring is adopted to minimize the one-hop broadcast count. By deferring, a transmitter will not broadcast messages immediately after receiving the beacon from the first-awake neighbor. In order to ensure that more neighbors receive the broadcast message, the

transmitter defers the broadcasting by $\delta = 1\ time\ slot$. By doing this, the first-awake neighbor can still receive the message, and the neighbors which wake up before the deferring time is due can also receive the broadcast message. Thus, deferring combines the advantages of opportunistic forwarding and the advantages of broadcasting over wireless radio.

As shown in Figure 6.1, suppose there are three neighbors for the transmitter. The transmitter only needs to broadcast two times (marked by the red arrow) to ensure that all neighbors will receive the message. This is more efficient than the pure opportunistic forwarding mechanism.

The only disadvantages of deferring is the additional latency (1 time slot for one-hop broadcasting) for flooding to the entire network. Therefore, deferring allows the tradeoff between the number of broadcast count and the broadcast latency to be exploited. We show in Section 6.3.2 that such additional latency is relatively small for the low duty-cycling case.

6.2.4 Online Forwarder Selection

In order to reduce the broadcast count or redundant transmission for multihop broadcasting, it is necessary to select as small number of relay nodes as possible. Many past efforts have formulated this problem as the Minimum Connecting Dominating Set (MCDS) problem [106]. However, we argue that a static MCDS cannot be applied for relay node selection in Hybrid-cast. First, to shorten the latency, it is necessary to select the relay nodes or forwarders along the direction of opportunistic forwarding, which results in online (or live) forwarder selection, rather than a static topology control as done in MCDS. Secondly, MCDS does not achieve minimum broadcast count in asynchronous duty-cycled WSNs due to multiple delivery for single hop broadcasting.

Algorithm 6: Algorithm for all node n_x:

set $N_{awake}(x)$;
$N^2_{reachable}(x) = \cup_{y \in N_{awake}(x)} N(y) - N_{awake}(x)$;
for $n_y \in N_{awake}(x)$ **do**
    if *node* $n_u \in N^2_{reachable}(x)$ *which is only reachable by* n_y **then**
        n_y is selected into O-$MPR(x)$;
        $N^2_{reachable}(x) = N^2_{reachable}(x) - n_u$;
while $N^2_{reachable}(x) \neq \emptyset$ **do**
    for $n_u \in N_{awake}(x)$ - *O*-$MPR(x)$ **do**
        n_m = n_u which covers the most nodes in $N^2_{reachable}(x)$;
    n_m is selected into O-$MPR(x)$;
    $N^2_{reachable}(x) = N^2_{reachable}(x)$ - node set covered by n_m;

In Hybrid-cast, initially, each node maintains its one hop awake neighbors (defined as $N(x)$) and the set of two hop neighbors $N^2(x)$ based on any underlying neighbor discovery protocols. The sink node or a relay node n_x computes the least number of relay nodes among its one-hop awake neighbors (defined as $N_{awake}(x)$) to cover the reachable two hop neighbors (defined as

$N^2_{reachable}(x)$).

$$N^2_{reachable}(x) = \cup_{y \in N_{awake}(x)} N(y) - N_{awake}(x) \tag{6.1}$$

The main purpose of the online forwarder selection algorithm in a transmitter n_x is to compute $N^2_{reachable}(x)$ as shown in Equation 6.1, and to compute the minimum number of relays to cover $N^2_{reachable}(x)$.

We adopt a heuristic solution, which is similar to the minimum multipoint relays (MPR) algorithm in [107]. The MPR problem is NP-Complete as shown in [107]. Thus, the minimum online forwarder selection problem is also NP-Complete. We denote the online MPR set for the transmitter n_x as O-$MPR(x)$. We provide a heuristic algorithm for computing O-$MPR(x)$ as described in Algorithm 6. An illustration is given in Figure 6.2(a).

Let us define the delivery latency from node n_i to node n_j as the time between when the data is ready in n_i and time at which the broadcast data is received by the neighbors, and denote the latency as $\tau_{i,j}(t)$ at time t ($\tau_{i,j}(t)$ is varying at different time). We have the following property.

Theorem 28 *Suppose node n_i has two neighbor n_j and n_k which are one hop away from each other. Then, at a time instant, t_i, we have the triangular property:*

$$\tau_{i,j}(t_i) \leq \tau_{i,k}(t_i) + \tau_{k,j}(t_i + \tau_{i,k}(t_i)) \tag{6.2}$$

Proof 22 *Suppose at time t_i, the data arriving time slot at n_j is t_j, and the data arriving time at n_k is t_k.*

If $t_k \leq t_j$, which means that the data arriving time at n_k is earlier than the data arriving time at n_j, $\tau_{i,k}(t_i + \tau_{k,j}(t_i + \tau_{i,k}(t_i)) = t_k - t_i + t_j - t_k = t_j - t_i = \tau_{i,j}(t_i)$. Otherwise, if $t_k > t_j$, which means that the data arriving time at n_k is later than the data arriving time at n_j, we have $\tau_{i,k}(t_i) + \tau_{k,j}(t_i + \tau_{i,k}(t_i)) = t_k - t_i + t_j^{'} - t_k > t_j - t_i + t_j^{'} - t_k > t_j - t_i > \tau_{i,j}(t_i)$. The theorem follows.

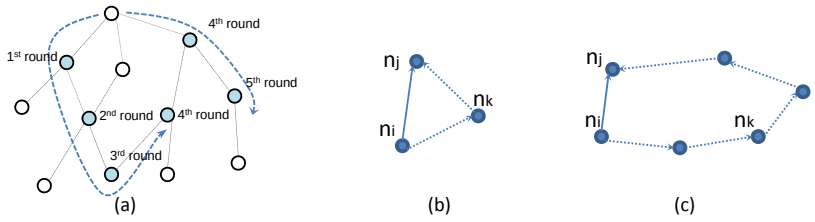


Figure 6.2: Online forwarder selection and the triangular path condition.

Theorem 28, as illustrated in Figure 6.2(b), illustrates that node n_i will always broadcast data to its one-hop neighbor n_j directly, without through other nodes.

We also have the following property.

Lemma 6 Triangular Path Condition*: For a node n_i and its neighbor n_j, at any time, the one-hop broadcast latency $n_i \rightarrow n_j$ is always the minimum possible.*

We omit the proof for the triangular path condition since it is a simple extension from that of Theorem 6.2. An illustration is given in Figure 6.2(c). Note that the triangular path condition does not exist in static networks. The triangular path condition indicates that the one-hop direct broadcast always achieves the least latency, in adaptively duty-cycled WSNs.

6.3 Performance Analysis

We now analyze the performance of Hybrid-cast in terms of the broadcast count and the broadcast latency, in order to illustrate its design advantages.

6.3.1 Upper-Bound on One-Hop Broadcast Count

We consider two scenarios in analyzing the one-hop broadcast count. In the low duty-cycling scenario, the schedule for a node n_i is waking up once every L_i time slots. In the quorum duty-cycling scenario, the schedule for a node n_i is waking up q times for every L_i consecutive time slots, where q is the quorum size.

Lemma 7 *[low duty-cycling] In Hybrid-cast, for a node n_i, the broadcast count is at least one, and at most $\max\{\Delta, L_m\}$, where Δ is the node degree of n_i and L_m is the maximum cycle length of nodes in the neighborhood.*

Proof 23 *If all nodes wake up within the same time slot, then after broadcast deferring, the transmitter can hear all neighbors, and one broadcast can cover all neighbors.*

Otherwise, if $\Delta \geq L_m$, the transmitter can hear all neighbors via staying awake for L_m time slots. Therefore, the maximum broadcast count is L_m. If $\Delta < L_m$, the transmitter can hear neighbors for at most Δ times, and the maximum broadcast count is Δ. Thus, the maximum number of broadcast count is $\max\{\Delta, L_m\}$.

By the Lemma 7, the upper bound of broadcast count in Hybrid-cast is at most n (where n is the network size) in the ideal case.

Lemma 8 *[quorum duty-cycling] In Hybrid-cast, for node n_i, the broadcast count is at least one, and at most $\max\{\Delta, q_m\}$, where Δ is the node degree of n_i and q_m is the largest quorum size of the quorum systems adopted by nodes in the neighborhood.*

Proof 24 *If all nodes wake up within one time slot, then after broadcast deferring, the transmitter can hear all neighbors, and one broadcast can cover all neighbors.*

Otherwise, if $\Delta \geq q_m$, the transmitter can hear all neighbors via staying awake in time slots scheduled by the quorum design. Therefore the maximum broadcast count is q_m. If $\Delta < q$, the transmitter will hear neighbors for at most Δ times, and the maximum broadcast count is Δ. Thus, the maximum broadcast count is $\max\{\Delta, q_m\}$.

6.3.2 Delivery Latency

Lemma 9 *Suppose the depth of the network (i.e., maximum layers by breadth-first-search) is D_{max}. Then, the upper bound for delivery latency is $L_m * D_{max} * T_s$ in low duty-cycling mode, where L_m is the maximum cycle length of nodes in the network. The upper bound is $q_m * D_{max} * T_s$ for quorum duty-cycling mode, where q_m is the largest quorum size of the quorum systems adopted by all nodes in the network.*

Proof 25 *Based on the Triangular Path Condition in Lemma 6, a node always broadcasts a message to its one-hop neighbors directly. Thus, for one hop broadcasting, the latency is at most $L_m * T_s$. After $L_m * T_s$, all nodes in the first layer will receive the broadcast message. Therefore, after $L_m * D_{max} * T_s$ time, all nodes in the network will receive the broadcast message.*

6.4 Simulation Results

We simulated Hybrid-cast using the OMNET++ simulator [102] and compared it against ADB [40] and opportunistic broadcasting [51] (denoted as OppFlooding).

Our experimental settings were consistent with the configurations in [18, 51]. We set the wireless loss rate as 0.1 and the duration of one time slot as 100 ms. The wireless communication range was set to 10m. We adopted the wireless loss model in [103], which considers the oscillation of radio. The size of the broadcast message packets was fixed as 512 bytes.

We examined the two main factors that affect the performance of our algorithms, including network size and duty-cycle setting. We generated a network with different number of nodes. For each network size, we randomly generated 10 topologies. Each data point reported in this section is the average of 10 topologies, with 10 runs on each topology. We varied the network size to understand its impact on the broadcast count and broadcast latency.

We measured the performance of the algorithms in a variety of duty cycle settings. For the low duty-cycling scenario, we varied the duration of the total periodic cycle length from $2T_s$ to $10T_s$ to generate heterogenous duty-cycling in a network for different nodes. For the quorum duty-cycling case, we choose the $(7, 3, 1)$, $(13, 4, 1)$, and $(21, 5, 1)$ difference sets for the heterogenous schedule settings. Since ADB, OppFlooding, and Hybrid-cast are independent of wakeup scheduling, we argue that the comparison is fair, even though ADB and OppFlooding do not explicitly support quorum duty-cycling.

6.4.1 Broadcast Count

We first measure the broadcast count which is the total number of broadcasting for flooding a message to the whole network. In this set of experiments, the network size was fixed by 200 nodes. The experimental results for different protocols are shown in Figure 6.3(a). In the low duty-cycling

case, Hybrid-cast outperforms ADB by approximately 50%, because of the less number of unicasts involved, due to the protocol's deferring and online forwarder selection.

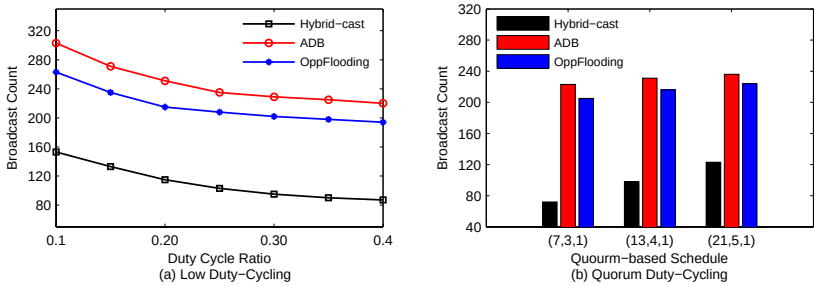


Figure 6.3: Performance comparison on broadcast count

For the quorum-based duty cycle setting, all nodes in the network chose homogenous quorum schedules. The setting was varied simultaneously for all nodes in different set of experiments. As shown in Figure 6.3(b), Hybrid-cast performs better since broadcasting are aggregated within quorum slots in each cycle. For example, for the $(7, 3, 1)$ setting (i.e., a node will stay awake at the 1^{st}, 2^{nd}, and 4^{th} slot on every 7 consecutive slots), there are at most 3 broadcasts to ensure that all neighbor nodes receive the broadcast message. However, for ADB and OppFlooding, the average one-hop broadcast count was 5 or 6, given the average degree in the network that we configured. The results validate the performance analysis in Section 6.3.1.

6.4.2 Broadcast Latency

Figure 6.4(a) shows the broadcast latency (defined as the time from broadcast beginning to all nodes receiving the broadcast data). With deferring, Hybrid-cast has slightly higher latency than ADB and OppFlooding, by about 10%, when the duty cycle ratio is 0.4, and by about 5%, when the duty cycle ratio is 0.1. As shown in Figure 6.4(a), as the duty cycle ratio decreases, the disadvantages of Hybrid-cast become more negligible, since the broadcast latency is more dominated by neighbor discovery latency.

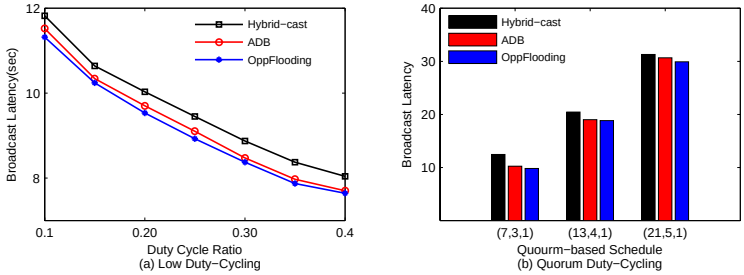


Figure 6.4: Performance comparison on broadcast latency

For the case of quorum duty-cycling, as shown in Figure 6.4(b), we observe a similar trend as that of low duty-cycling. The latencies for all three protocols tend to increase with larger quorum cycle. However, the latencies tend to converge to the same value when the quorum cycle increases. This is because, the neighbor discovery latency is approximately linearly increasing with quorum cycle, as shown in [1] The results also validate the performance analysis in Section 6.3.2.

6.4.3 Impact of Network Size

We also evaluated the impact of network size and heterogenous duty-cycling on message count and broadcast latency. For the low duty-cycling case, each node randomly selected a duty cycle ratio in the range 0.1 to 0.4. For the quorum duty-cycling case, we chose the $(7, 3, 1)$, $(13, 4, 1)$, and $(21, 5, 1)$ difference sets for the schedules of all nodes (the non-empty intersection property among these sets was proved in [1]). In the simulation experiments, we varied the network size from 200 nodes to 1600 nodes.

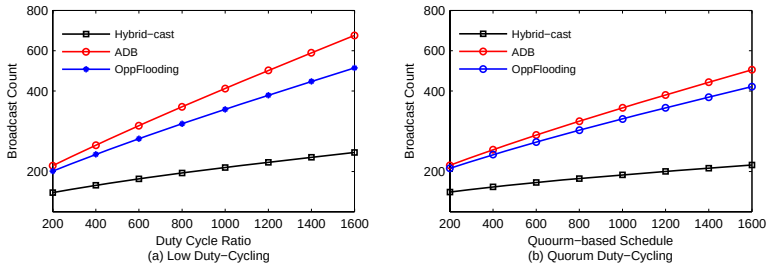


Figure 6.5: Broadcast count with different network size.

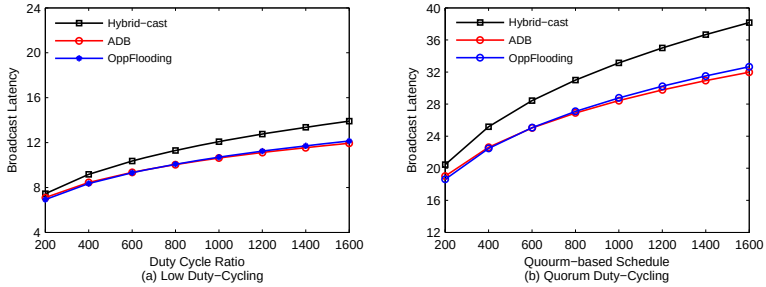


Figure 6.6: Broadcast latency with different network size.

As shown in Figure 6.5, as the network size increases, the message count of Hybrid-cast and the other two solutions exhibit an increasing trend. This is because, more relay nodes will be selected in larger networks. The same trend exists for broadcast latency as shown in Figure 6.6, as there are more hops along the breadth-first-search tree. This is consistent with the analysis in Section 6.3.2.

We also evaluated the impact of network size for quorum-based duty cycle setting. We observed similar trends for the broadcast count and broadcast latency as that in the low duty-cycling setting.

The performance comparisons thus illustrate the performance tradeoff achieved by Hybrid-cast.

6.5 Discussions and Conclusions

Note that we do not assume local synchronization or duty-cycle awareness, which is required by past works such as [51] and [83]. The assumption in Hybrid-cast is neighbor-awareness. Such awareness can be achieved by neighbor discovery protocols, or by quorum-based duty-cycling [1]. Each node will inform its neighbors after reconfiguration on the duty-cycling.

By adopting quorum duty-cycling, Hybrid-cast can be extended to mobile WSNs, because neighbor discovery is guaranteed within bounded time in quorum duty-cycling, as shown in [1].

Due to the problem of hidden terminal, it is possible that one node may receive broadcast messages from two nodes simultaneously, which leads to collision. For reliable broadcasting, if a node received the broadcast, it can set a mark field in the beacon message. By checking the beacon message from the neighbor, a transmitter can decide whether retransmission is necessary. We do not defer broadcast for retransmission. The transmitter could backoff a random period $0 \leq t \leq T_s$ in order to avoid collision.

We do not explicitly consider reliability issues in Hybrid-cast. However, the traditional ACK and NACK mechanisms for reliable data transmission can be applied to Hybrid-cast to support reliable broadcasting.

In this chapter, we designed an asynchronous broadcasting protocol, Hybrid-cast, for WSNs with adaptively low duty-cycling or quorum-based duty-cycling schedules. The main difficulty of this problem is that, sensor nodes are not time-synchronized and do not stay awake simultaneously. Hybrid-cast broadcasts messages to the neighbors who wake up early, in order to shorten the broadcast latency. Previous solutions often use multiple unicasts for broadcasting, which incurs high overhead. To overcome the disadvantages of such multiple unicasts, Hybrid-cast defers broadcasting to ensure that the number of awake neighbors is as large as possible. We also selected the minimum relay points online in order to reduce broadcast count and collisions.

We mathematically established the upper bound of broadcast count and broadcast latency for a given duty-cycling schedule. We compared the performance of Hybrid-cast with ADB and OppFlooding protocols. Our simulation results validated the effectiveness and efficiency of our design.

Chapter 7

Prototype Implementations

We implemented a prototype system over Telosb [10] from Crossbow Technology and TinyOS 2.0 distribution [93] to validate our algorithms and protocol design for asynchronous quorum-based wakeup scheduling and neighbor discovery. We targeted to the application of real-time target tracking/monitoring over duty-cycled WSNs.

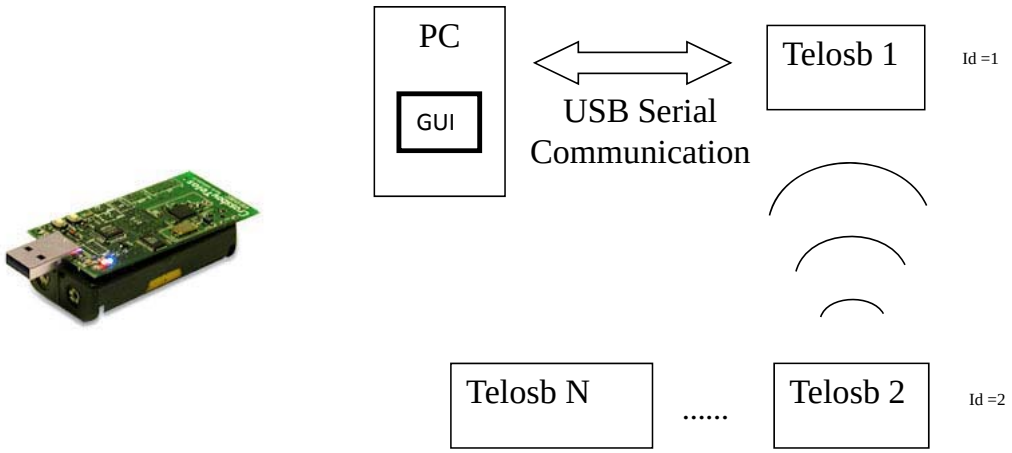
We changed the protocol stack of TinyOS [93] by integrating new protocols proposed by us into the operating system. Two important metrics were measured for evaluation from the implementation: event-to-sink delay which indicates time between when a sentry node discovers a target and the time at which a report arrives at sink node; and energy saving ratio which represents the ratio of wakeup time to entire demonstration duration.

7.1 Hardware and Software Platform

Each node in the system adopted Crossbow TPR2400/2420 ("Telosb") Mote [10] platform. Telosb is an open-source experimental platform developed and published by the University of California, Berkeley ("UC Berkele"). The platform provides an integrated processor (TI MSP430 microcontroller) radio solution including a USB interface, 2.4 GHz radio, PCB antenna, sensor interfaces. The Telosb is the same as the TPR2400 except that it comes with a pre-installed environmental sensor suite. It is a research platform upon which cutting edge experiments on new ideas in sensor networks can be developed. The IEEE 802.15.4 compliant radio support built into the Telosb allows for communications with other 802.15.4 enabled platforms, including the commercial grade FCC/ARIB certified MICAz Mote. As an uncertified radio platform, the TPR2400/2420 intended for research and lab studies. The Telosb is fully compatible with UC Berkeleys open-source TinyOS distribution [93].

We deployed almost 20 nodes in the prototype system development. The Figure 7.1 show the Telosb device we used and the architecture in the proto-type network system.

We programmed over Telosb via *Cygwin* environment. Basically, the development contains con-



(a) Telosb (b) Software Architecture

Figure 7.1: Telosb and System Architecture in Implementations

figuration and programming. The main steps of programming a TelosB Mote through PC USB interface is as follows: (1) Identify the COM port: Plug one TPR2400/2420 into a PC; Open a Cygwin shell and type motelist. Then this command will find the COM port your TPR2400/2420 is on; and (2) Compile the TinyOS Application: Change your directory to the CntToLedsAndRfm directory (inside a Cygwin window); Compile (type 'make telosb'); and (3) Compile the program: type "make telosb reinstall,$\langle node\_ID \rangle$ bsl, $\langle N \rangle$" in he Cygwin window, where $\langle node\_ID \rangle$ sets the node address (optional) and $\langle N \rangle$ is an integer one less than the COM port number assigned to the TPR2400/2420. For example if the TelosB is assigned COM port 11, then the value of $\langle N \rangle$ would be 10.

We did not implement the sensing function and did not contain any sensing unit in the TelosB device. In order to simulate the sensing and reporting process, we used a mobile TelosB to act as a "target", which we referred as to the target node. The target node sent out radio periodically (once every 200 mill seconds). When a common sensor nodes detected the radio from the target nodes, it would send a report towards the sink node.

There was one sink node which communicated through USB port with a PC that acts as a central monitoring station. Other nodes, when detected the moving target, reported a message with their own location to the sink node. When the sink node received the reports, it forwarded the message to the PC monitoring software, where the location of the target was shown. The GUI of the monitoring software is shown in Figure 7.2. The software which is developed over Windows by Visual Studio 2005 contains 3 parts: target location area, performance area, and physical onsite monitoring.

Regarding the TinyOS [93], it is an open-source operating system designed for wireless embedded sensor networks. The main feature of TinyOS is the component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in WSNs. TinyOS's component library includes network protocols, distributed

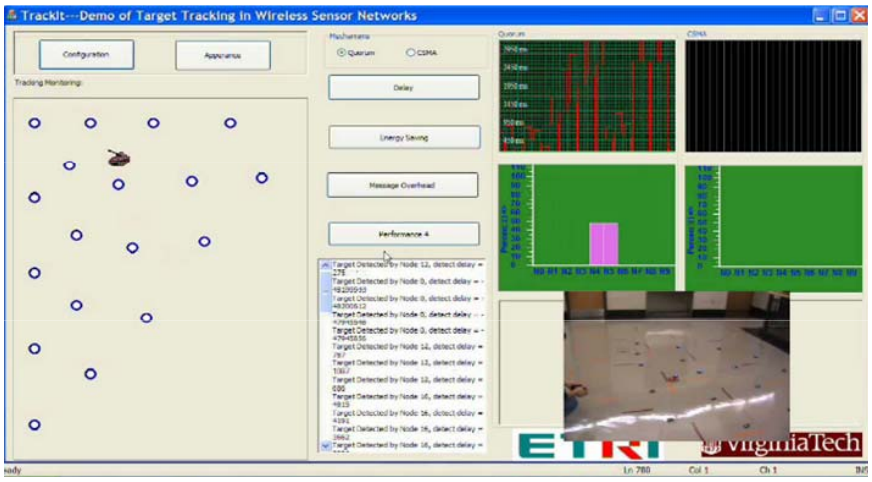


Figure 7.2: Monitoring GUI on PC

services, sensor drivers, and data acquisition tools – all of which can be used as-is or be further refined for a custom application. TinyOS's event-driven execution model enables fine-grained power management yet allows the scheduling flexibility made necessary by the unpredictable nature of wireless communication and physical world interfaces.

Armed with the open-source OS, we did not implement everything from the scratch. The major works was done for protocol stacks modification, which is described in detail in Section 7.2.

7.2 Protocol Stacks and Modules

The implementation is based TinyOS 2.0 protocols stack. Our implementation is cross MAC layer and Application layer, which means the main functionality is finished in the MAC layer but the configuration interface is done in the application layer, as shown in Figure 7.3.

The underlying routing protocol is configurable, it can be MintRoute, DSDV or a static route configuration given a small-sized network. The MAC protocol can be IEEE 802.15.4 or CSMA-based protocol. In our prototype implementation, we choose CSMA upon which our quorum-based wakeup scheduling is built. Meanwhile, it is easy to investigate the energy efficiency of our solution by comparing with CSMA which is an always-awake solution.

The main module we implemented is the *QuorumWakeupP* interface which is inherited from *PowerCycle* interface and *SplitControl* interface. It contains the following functions:

1. task void Initiation();
2. task void stopRadio();
3. task void startRadio();
4. task void getCca();

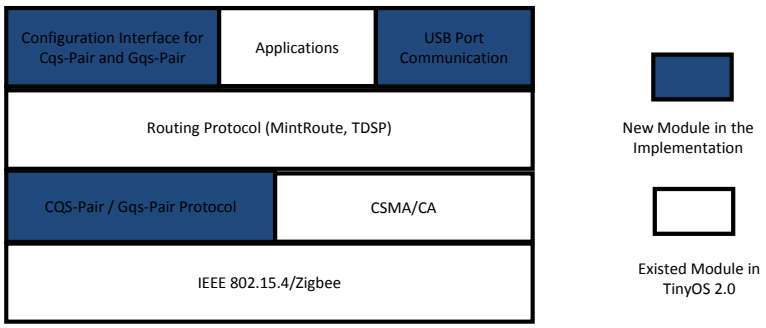


Figure 7.3: Protocols stacks in the prototype system

5. bool finishSplitControlRequests();
6. bool isDutyCycling();
7. message_t* receive(message_t* msg, void* payload, uint8_t len);
8. task void radioSendTask();

We set the time duration of one time slot through the `getSleepInterval()` command and `setSleepInterval(uint16_t sleepIntervalMs)` command in the `PowerCycle` component. The state machine in our implementation will be illustrated in Section 7.3.

7.3 Implementations of cqs-pair and gqs-pair

There are three key issues in converting the cqs-pair and gqs-pair concepts into practical implementations. The first key issue is to ensure that two nodes can discover each other in the presence of clock drift. The second one is that a node should keep awake if there is pending data for receiving or for transmitting. The third issue is how to support multicast or broadcast.

7.3.1 Beacon Messages

Previous work on the implementation of QPS protocol over IEEE 802.11 adopts the concept of ATIM (Ad hoc Traffic Indication Map) windows [11], in which a node can optionally enter the sleep mode if it receives no ATIM frame in an ATIM window.

In our implementation, we do not use the notion of ATIM windows. We define the time interval that a node is scheduled to be awake as an *active slot*, and the time interval that the node is scheduled to sleep as a *silent slot*. In an *active slot*, a node has to transmit its own *beacon message* to inform its neighbors about its wakeup status, and listen to *beacons* from other nodes for which it may have buffered packets that are waiting for transmission.

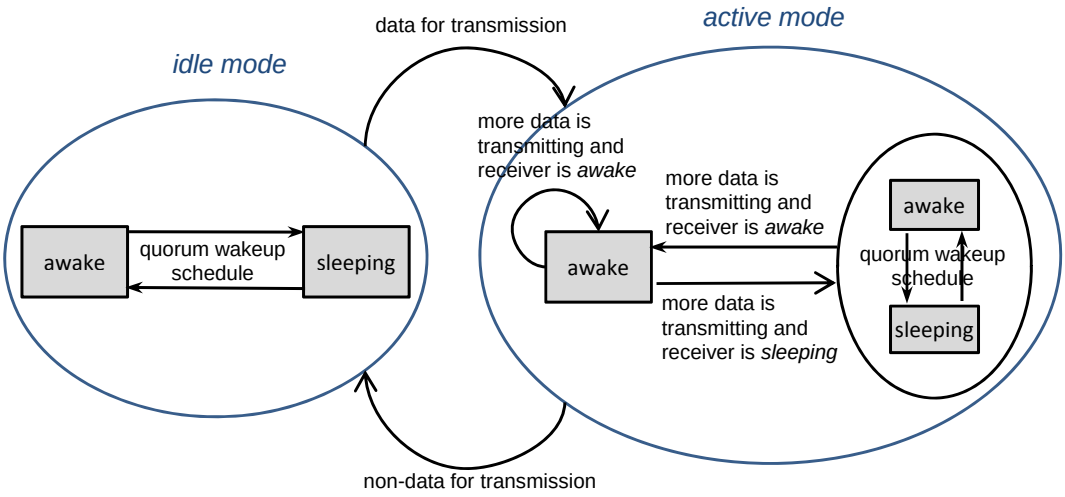


Figure 7.4: Power Management at the Transmitter Side: Communication Schedule and Wakeup schedule

In our scheme, to ensure the correctness of the protocol, a node remains awake throughout its entire active slot. It may be possible for nodes to be only partially awake during their active slots – such optimizations can be considered in future works. In a *silent slot*, a node will shut down its radio.

The beacon message contains three fields:{*indic, node_id, time_stamp*}. The *indic* field can have only two types of value: indic=0, which indicates that the message is a beacon message, and indic = 1, which indicates that the message is data. The *node_id* field is used to distinguish among different nodes. The *time_stamp* field is used to identify whether or not two beacon messages are identical.

7.3.2 Power Management

The goal of power management is to facilitate effective communication while saving as much energy as possible. In our power management scheme, a node determines its desirable *communication schedule*, i.e., when it should go to sleep or wake up. The relationship between the wakeup schedule and the communication schedule devised by a power management policy for a sender is illustrated in Figure 7.4.

At the MAC-layer, we propose a reservation mechanism for communication on top of the proposed quorum-based heterogenous wakeup scheduling scheme (cqs-pair or gqs-pair). Each node has two states, *idle mode* and *active mode*. In the *idle mode*, a node will follow its wakeup schedule to wake-up or to sleep. We also call this mode as power saving mode. Once there is data for receiving or for transmitting, the node will enter into the *active mode* as shown in Figure 7.4.

In the active mode, a sender maintains a table of timers for all its neighbors. The timers are triggered once the sender receives beacon messages from the neighbors. The initial value of each timer is one time slot. The sender will also record its own wakeup schedule via a timer. If both the sender and the receiver is in an *active slot*, then they can communicate. If the sender enters into a *silent slot* but there are more packets for transmission and the receiver is still in an *active slot*, then the sender will keep awake in its next slot. If there are more packets for transmission, but the receiver will enter a *silent slot*, then the sender will send a *keep-awake* message to the receiver at the end of the transmission of the current packet. The receiver that is being requested to stay awake will then send back an acknowledgment, indicating its willingness to remain awake in its next slot.

The power management scheme at the receiver side is simpler than that at the sender side. In active mode, if there is no *keep-awake* message, the receiver will continue communication until the end of its current slot interval; otherwise, it will keep awake in its next slot.

7.3.3 Support for Multicast and Broadcast

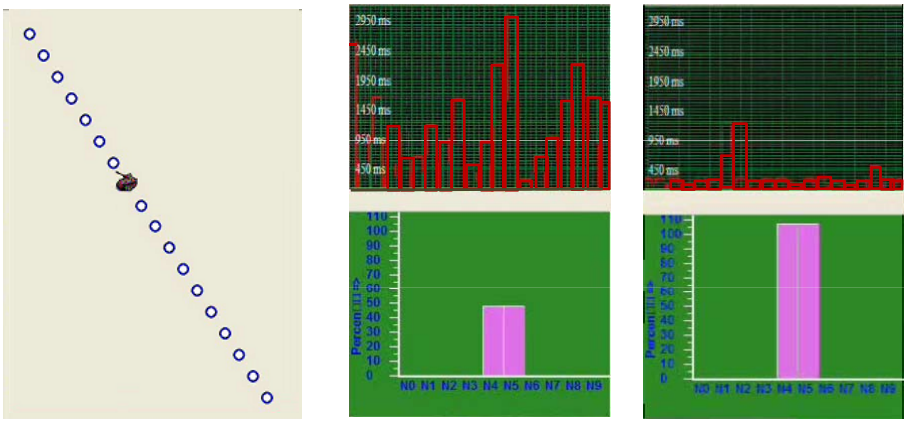
Quorum-based asynchronous wakeup protocols cannot guarantee that more than one receiver is awake when a sender wishes to multicast or broadcast.

There are multiple ways to support multicast and broadcast. One method is to adopt relatively prime frequencies among all nodes for wakeup scheduling. This method does not need synchronization between the sender and all the receivers. The sender only needs to notify m receivers to wake-up via the pairwise relative primes p_1, p_2,..., p_m, respectively. Then each receiver generates its new wakeup frequency based on the received frequency. Through Chinese Remainder Theorem [68, 88], it can be proven that the m receivers must wakeup simultaneously at the I^{th} beacon interval ($0 \leq I \leq p_1 \times p_2 ... \times p_m$). The sender can then transmit a multicast/broadcast message at this interval.

Another way to multicast/broadcast is by using synchronization over quorum-based wakeup schedules. The sender can book-keep all neighbors' schedules, and synchronize their schedules so that neighboring nodes wake up in the same set of slots with the use of Lamport's clock synchronization algorithm [108]. When all nodes are awake simultaneously, the senders then send a message to multiple neighbors simultaneously.

The first mechanism has the advantage that no synchronization is needed between a sender and multiple receivers. But it cannot bound the average delay. The second approach can bound the average delay but it needs book-keeping and synchronization over asynchronous wakeup schedules.

For multicast/broadcast, we set a threshold L. If the number of multicast packets exceed the threshold L, the sender will send a Multicast-Notify to all neighbor receivers, requesting them to stay awake. Otherwise the sender will send the multicast data to each receiver one by one, by unicasting. The value of the threshold L depends on the configuration of time slot lengths and packet lengths.



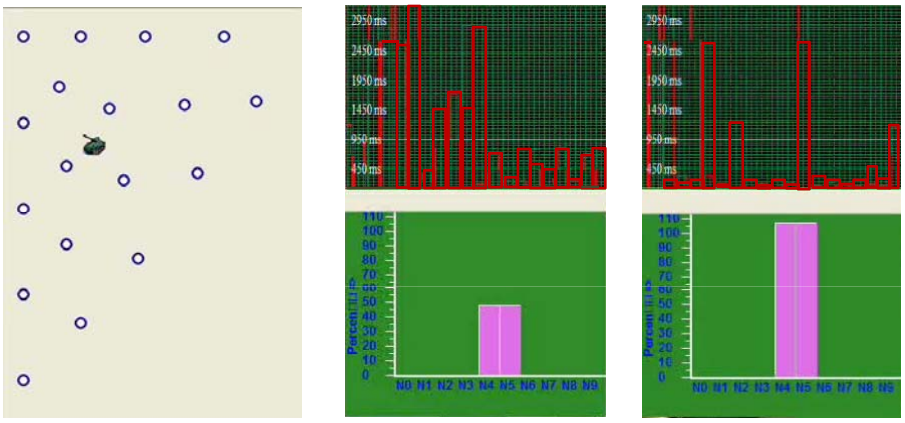
(a) Line Topology

(b) Performance Comparison (Upper:End-to-end latency; Below:Energy consumption ratio)

Figure 7.5: Performance Comparison for Line Topology

To reduce the time of waiting before actual transmission, the Multicast-Notify message contains a field to notify all receivers on how long they should wait. The value of this field is the time between when the message is sent and when all receivers are awake.

7.4 Demo Performance



(a) Tree Topology

(b) Performance Comparison (Upper:End-to-end latency; Below:Energy consumption ratio)

Figure 7.6: Performance Comparison for Tree Topology

In the proto-type implementation, performance regarding energy saving ratio and e2e latency were compared between Quorum-based wakeup scheduling and CSMA protocol. The application run-

ning was target tracking. We adopt two topologies for the comparisons: line topology and tree topology. We show below the live performance comparison in our demonstration before.

From the demonstration with both line topology and tree topology, we observe that there is a tradeoff between the two mechanisms. As shown in Figure 7.5, although Quorum-based wakeup scheduling achieved less energy consumption ratio, but it suffered higher latency compared with CSMA protocol.We set the time slot length as 100ms in the Quorum-based wakeup scheduling, and select the (7,3,1)-difference set design. When the hop count is 8, the end-to-end latency is almost over 2000ms for the Quorum scheme, but only about 500ms for CSMA protocol. However, the Quorum scheme only stay awake 42.8% time in idle state, comparing 100% for CSMA protocol. The similar trend existed for tree topology, as shown in Figure 7.6.

The demonstration performance verifies the design trade-off in the cqs-pair and gqs-pair protocols. Thus, when saving energy is an critical issue for sensor nodes, we should sacrifice the latency in order to prolong the node lifetime individually.

Chapter 8

Conclusions and Future Work

This dissertation is motivated by saving idle energy consumption, providing optimal data routing in terms of E2E latency, and supporting efficient multihop broadcasting in duty-cycled WSNs. The work is based on the observation that idle energy consumption for sensor nodes is not negligible for a large class of WSN applications (e.g., environment monitoring [4], target tracking [5]). The dissertation adopts the duty-cycling approach for saving idle energy. However, this raises new challenges on neighbor discovery, data delivery with optimal latencies, and data broadcasting. Toward this, the dissertation focuses on, and solves three problems: a) asynchronous wakeup scheduling and the corresponding neighbor discovery mechanism, b) time-varying shortest path routing due to varying link costs, and c) efficient multihop broadcast.

Our solutions on asynchronous wakeup scheduling were inspired by the quorum-based power saving mechanism [11]. We proposed two designs, cqs-pair and ggs-pair, and provide a general framework in which we can satisfy the heterogenous energy saving requirement, while guaranteeing neighbor discovery latency. We show that this generalization is a non-trivial extension from past works.

Armed with the design rationale (described in Chapter 4) of quorum systems, we first proposed cqs-pair, which guarantees that two adjacent nodes which adopt heterogenous cyclic quorums from such a pair as their wakeup schedules, can hear each other at least once within one super cycle length (i.e., the larger cycle length in the cqs-pair). We also developed a fast algorithm for constructing cqs-pairs, using the *multiplier theorem* [44] and the (N, k, M, l)-difference pair defined by us. With the help of the cqs-pair, WSNs can achieve better trade-off between energy consumption and average delay. For example, all cluster-heads and gateway nodes in a heterogenous WSN can select a quorum from the quorum system with smaller cycle length as their wake up schedules, to obtain smaller discovery delay. In addition, all members in a cluster can choose a quorum from the quorum system with longer cycle length as their wakeup schedules, in order to save more idle energy. cqs-pair is the first asynchronous wakeup scheduling algorithm for heterogenous energy saving and meanwhile guaranteeing bound neighbor discovery latency over WSNs.

Another design that we propose on heterogenous quorum-based wakeup scheduling is gqs-pair, in

which each quorum system of the pair is a grid quorum system [45]. We prove that any two grid quorum systems can form a *gqs-pair*. Compared with cqs-pair, gqs-pair has better performance in terms of average neighbor discovery latency. With the help of the gqs-pair, WSNs can also achieve better tradeoff between energy consumption and average delay. Meanwhile, gqs-pair has better performance in terms of average neighbor discovery latency than cqs-pair.

We see from cqs-pair and gqs-pair that the quorum design can be applied as a framework for asynchronous wakeup scheduling in order to save energy in heterogenous WSNs. The framework can achieve tradeoff between energy saving ratio and discovery latency, and hence provide configuration flexility for different applications which are either energy-critical or latency-critical.

We also considered asymmetric quorum design to further improve the energy efficiency in two-tiered WSN topologies. We developed a flexible solution, p-Grid, for neighbor discovery service in low duty-cycled WSNs. The motivation behind p-Grid was to provide an energy-efficient, run-time configurable, asynchronous neighbor discovery mechanism for WSNs in unreliable environments. p-Grid is easy to implement: in idle state, nodes select a read quorum from a prime-grid quorum group as their low power listening schedules. For neighbor discovery, a node switches from listening mode to probing mode by selecting a write quorum from a prime-grid quorum group as its probing schedule, and sends out DISC messages in the time slots selected as a write quorum. Armed with the prime-grid quorum group design, there must be at least one overlapping slot in which both the probing node and its neighbors in listening mode are awake simultaneously.

Therefore, p-Grid achieves better energy efficiency for neighbor discovery in unreliable environments, compared with conventional protocols. We conclude from p-Grid protocol that asymmetric design on asynchronous wakeup scheduling can further improve the energy efficiency when there is no data communication between cluster members (which is true for most of WSNs applications, like target tracking [5] and environment monitoring [4]).

The designs of cqs-pair, gqs-pair, and p-Grid constitute the first contribution in the dissertation.

The second contribution of the dissertation is time-dependent shortest path routing in duty-cycled WSNs. We model duty-cycled WSNs as time-dependent networks, which satisfy the FIFO condition. We then developed the FTSP distributed algorithm for finding shortest paths in such networks. FTSP has polynomial message complexity and is more time-efficient than previous solutions. We also developed the FTSP-M algorithm for distributed route maintenance with node insertion, configuration updating, and deletions. FTSP-M is memory efficient and has polynomial message complexity. Finally, we developed a sub-optimal implementation of FTSP and FTSP-M in order to reduce the vector size. The vector size of the sub-optimal implementation does not depend on the largest LCM value as shown in Equation 5.6.

Comparing with past works [50], the algorithms proposed by us have lower time and message complexities in the worst case. And the number of execution times for enumerating shortest paths for all discrete time moments is bounded.

Our work on this part indicates that adaptively duty-cycled WSNs exhibits an unique time-dependent feature: periodically varying link costs. Based on the periodical feature, the construction of

time-dependent shortest paths can be expedited largely compared with that for traditional time-dependent networks.

The dissertation's third contribution is our solution for multihop broadcast in duty-cycled WSNs, which is an important service for many WSN applications, such as configuration update, message flooding, route discovery etc. Toward this, we proposed Hybrid-cast. The main difficulty in the design of a multihop broadcast protocol in duty-cycled WSNs is that, sensor nodes are not time-synchronized, and therefore, do not stay awake simultaneously. Hybrid-cast broadcasts messages to the neighbors who wake up early, in order to shorten the broadcast latency. Previous solutions often use multiple unicasts for broadcasting, which incurs high overhead. To overcome the disadvantages of such multiple unicasts, Hybrid-cast defers broadcasting to ensure that the number of awake neighbors is as large as possible. We also selected the minimum relay points online, in order to reduce broadcast count and collisions. We analytically established the upper bound of broadcast count and broadcast latency for a given duty-cycling schedule.

Comparing with previous solution (i.e., pure opportunistic flooding [51], and unicast replacement approach [40]), Hybrid-cast achieves better tradeoff between broadcast latency and broadcast count compared to previous broadcast solutions. Meanwhile, it reduces redundant transmission via delivery deferring and online forwarder selection.

Hybrid-cast protocol reveals that the broadcast count can be largely reduced with small sacrifice in broadcast latency which is critical for energy saving. Hence, some broadcast services without sensitive timeline requirement, such as code update and reconfiguration over duty-cycled WSNs, would benefit from it.

The dissertation's fourth and last contribution is the prototype system development over real hardware and software platform. We implemented cqs-pair and gqs-pair in a WSN platform comprised of Telosb [10] motes running TinyOS 2.0. The cqs-pair and gqs-pair designs were integrated with the existing protocol stacks in TinyOS 2.0. We measured the performance of the designs in terms of energy-saving ratio and neighbor discovery latency, to show the superiority of our designs. Our implementation did not modify the upper layer routing protocols, and is pluggable to the existing software architecture and protocol stack in TinyOS.

We conclude from the implementation that our solutions for quorum-based wakeup scheduling can be applied to real WSN platforms with desired performance, and that the implementation of those solutions can work well with existing protocol stack and software modules.

8.1 Summary of Contributions

To summarize, the dissertation's contributions include:

- We developed a set of asynchronous quorum-based wakeup scheduling schemes in duty-cycled WSNs including cqs-pair, ggs-pair, and p-Grid, and show that they provide a better

trade-off between energy consumption and average delay, and better energy efficiency for neighbor discovery in unreliable environments, compared with existing protocols.

- We modeled duty-cycled WSNs as time-dependent networks, and developed the FTSP and FTSP-M distributed algorithms for finding shortest paths in such networks. We show that FTSP and FTSP-M algorithms are superior to past solutions.
- We developed a multihop broadcast protocol called Hybrid-cast for broadcasting in duty-cycled WSNs. Hybrid-cast upper bounds the broadcast count and broadcast latency for a given duty-cycling schedule. We establish the superiority of Hybrid-cast over past solutions.
- We developed a prototype implementation of the cqs-pair and gqs-pair asymmetric quorum designs in a Telosb/TinyOS WSN platform. The implementation validate the feasibility over existing WSN platforms. The experimental measurements show the superiority of over designs over existing solutions.

8.2 Future Work

Based on the dissertation's results, we envision the following directions as promising avenues for further research.

One direction is to consider a general framework for adopting quorum systems in asynchronous wakeup scheduling. In the dissertation, we only discussed two quorum systems, cyclic quorum systems and grid quorum systems. However, it may be possible to design better quorum systems which can be utilized for designing asynchronous wakeup scheduling schemes in duty-cycled WSNs. A general framework for designing such schemes is desirable, which can provide a systematic view and deeper insights on quorum systems that can yield greater energy savings or optimized E2E latencies.

Another possible direction is capacity maximization. Although asynchronous quorum-based wakeup scheduling is energy efficient, it does not consider the collision during simultaneous transmission to same receivers, which can happen frequently during data aggregation. Such collisions can significantly degrade the capacity and throughput. Existing asynchronous mechanisms, such as RTS/CTS, are not efficient for this problem. Scheduled transmission is one promising solution, but it requires clock synchronization. Hence, a protocol which combines the merits of both asynchronous protocols and synchronized protocols (or a hybrid solution), or some improved versions with random quorum selection, or practical back-off time setting would be a promising line of research for capacity maximization in data aggregation WSN applications.

Several interesting future research directions also exist in the problem space of time-varying E2E data delivery in duty-cycled WSNs. Building upon the dissertation's solutions on the time-dependent shortest path routing problem, one can consider the time-dependent minimum spanning tree problem, which is NP-Hard, or consider improving existing routing protocols (i.e., minimum-hop routing) with time-varying optimal latencies. Another very interesting direction is the time-dependent

multicast routing problem, which is the reverse direction of all-to-one routing discussed in the dissertation. Last but not least, we believe that it is worthy of considering interference in the time-varying routing, which is probably one new research problem.

Bibliography

[1] S. Lai, B. Zhang, B. Ravindran, and H. Cho, "Cqs-pair: Cyclic quorum system pair for wakeup scheduling in wireless sensor networks." in *International Conference on Principles of Distributed Systems (OPODIS)*, vol. 5401. Springer, 2008, pp. 295–310.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[3] G. Platt, M. Blyde, S. Curtin, and J. Ward, "Distributed wireless sensor networks and industrial control systems - a new partnership," in *Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors (EmNets05:)*, Washington, DC, USA, 2005, pp. 157–158.

[4] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (WSNA02:)*, 2002, pp. 88–97.

[5] T. He, P. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, R. Stoleru, Q. Cao, J. A. Stankovic, and T. Abdelzaher, "Achieving real-time target tracking using wireless sensor networks," in *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS06)*, 2006, pp. 37–48.

[6] J. A. Stankovic, Q. Cao, T. Doan, L. Fang, Z. He, R. Kiran, S. Lin, S. Son, R. Stoleru, and A. Wood, "Wireless sensor networks for in-home healthcare:," in *Proceeding of High Confidence Medical Devices, Software, and Systems (HCMDSS05)*, 2005, pp. 2–3.

[7] K. Romer and F. Mattern, "The design space of wireless sensor networks," *Wireless Communications, IEEE*, vol. 11, no. 6, pp. 54 – 61, Dec. 2004.

[8] L. M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *IEEE Conference on. Computer Communications (INFOCOM)*, 2001, pp. 1548–1557.

[9] T. I. (TI), "Cc2420 data sheet," http://focus.ti.com/lit/ds/symlink/cc2420.pdf.

[10] Crossbow, "Telosb datasheet," http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf.

[11] C. H. Y. Tseng and T. Hsieh, "Power-saving protocols for ieee 802.11-based multi-hop ad hoc networks," in *IEEE Conference on. Computer Communications (INFOCOM)*, 2002, pp. 200 – 209.

[12] R. Zheng, J. C. Hou, and L. Sha, "Asynchronous wakeup for ad hoc networks," in *Proceedings of the ACM symposium on Mobile ad hoc networking and computing (MobiHoc)*, 2003, pp. 35–45.

[13] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems (Sensys)*, 2004, pp. 95–107.

[14] T. Voigt, H. Ritter, and J. Schiller, "Utilizing solar power in wireless sensor networks," in *Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks (LCN)*, 2003, p. 416.

[15] C. Vigorito, D. Ganesan, and A. Barto, "Adaptive control of duty cycling in energy-harvesting wireless sensor networks," in *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, June 2007, pp. 21–30.

[16] J. A. Paradiso and T. Starner, "Energy scavenging for mobile and wireless electronics," *IEEE Pervasive Computing*, vol. 4, no. 1, pp. 18–27, 2005.

[17] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Trans. Embed. Comput. Syst.*, vol. 6, no. 4, p. 32, 2007.

[18] R. J., P. B., and C. V., "Adaptive low power listening for wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, pp. 988–1004, 2007.

[19] C. Schurgers, S. G. V. Tsiatsis, and M. Srivastava, "Topology management for sensor networks: Exploiting latency and density," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2002.

[20] M. J. Miller and N. H. Vaidya, "Power save mechanisms for multi-hop wireless networks," in *Proceedings of the First International Conference on Broadband Networks (BROADNETS)*, 2004, pp. 518–526.

[21] L. Gu and J. Stankovic, "Radio-triggered wake-up capability for sensor networks," in *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2004, pp. 27–37.

[22] Y. Sun, S. Du, O. Gurewitz, and D. B. Johnson, "Dw-mac: a low latency, energy efficient demand-wakeup mac protocol for wireless sensor networks," in *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc08:)*, 2008, pp. 53–62.

[23] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 12, pp. 493–506, 2004.

[24] T. Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *The First ACM Conference on Embedded Networked Sensor Systems (Sensys)*, 2003.

[25] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare-event detection," in *Proceedings of the 4th international symposium on Information processing in sensor networks (IPSN)*, 2005.

[26] A. Keshavarzian, H. Lee, and L. Venkatraman, "Wakeup scheduling in wireless sensor networks," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, 2006, pp. 322–333.

[27] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks," in *Proceedings of the 4th international conference on Embedded networked sensor systems (Sensys)*, 2006, pp. 307–320.

[28] A. El-Hoiydi and J.-D. Decotignie, "Wisemac: an ultra low power mac protocol for the downlink of infrastructure wireless sensor networks," in *Proceedings of the Ninth International Symposium on Computers and Communications (ISCC)*, 2004, pp. 244–251.

[29] N. Vasanthi and S. Annadurai, "Aws: asynchronous wakeup schedule to minimize latency in wireless sensor networks," in *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on*, vol. 1, June 2006, pp. 7–.

[30] C. H. J.R. Jiang, Y.C. Tseng and T. Lai, "Quorum-based asynchronous power-saving protocols for ieee 802.11 ad hoc networks," *ACM Journal on Mobile Networks and Applications (MONET)*, 2005.

[31] I. Chou, C. Chao, and J. Sheu, "An adaptive quorum-based energy conserving protocol for ieee 802.11 ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 5, pp. 560–570, 2006.

[32] Z.-T. Chou, "A randomized power management protocol with dynamic listen interval for wireless ad hoc networks," in *Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd*, vol. 3, 2006, pp. 1251–1255.

[33] O. Younis and S. Fahmy, "Heed: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.

[34] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)*, 2004, pp. 214–226.

[35] A. Amis and R. Prakash, "Load-balancing clusters in wireless ad hoc networks," in *Proceedings of 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology*, 2000, pp. 25–32.

[36] T. Hou and T. Tsai, "A access-based clustering protocol for multihop wireless ad hoc networks," *Selected Areas in Communications, IEEE Journal on*, vol. 19, no. 7, pp. 1201–1210, Jul 2001.

[37] C. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 15, no. 7, pp. 1265–1275, Sep 1997.

[38] O. Younis and S. Fahmy, "Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 3, no. 4, pp. 366–379, Oct.-Dec. 2004.

[39] Y. Gu and T. He, "Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links," in *Proceedings of the 6th ACM conference on Embedded network sensor systems (Sensys)*, 2007, pp. 32–38.

[40] Y. Sun, O. Gurewitz, S. Du, L. Tang, and D. B. Johnson, "Adb: an efficient multihop broadcast protocol based on asynchronous duty-cycling in wireless sensor networks," in *ACM SenSys*, 2009, pp. 43–56.

[41] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *5th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom)*, 1999, pp. 151–162.

[42] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing (MobiHoc)*, 2002, pp. 194–205.

[43] S. Lai and B. Ravindran, "On distributed time-dependent shortest paths over duty-cycled wireless sensor networks," in *IEEE Conference on. Computer Communications (INFOCOM)*, 2010.

[44] D. R. Stinson, *Combinatorial Designs: Constructions and Analysis*. SpringerVerlag., 2003.

[45] W. Luk and T. Huang, "Two new quorum based algorithms for distributed mutual exclusion," in *International Conference on Distributed Computing Systems (ICDCS)*, 1997, pp. 100 – 106.

[46] S. Lai, B. Ravindran, and H. Cho, *IEEE Transaction on, Computers (TC)*, vol. 60.

[47] D. P. Bertsekas, "A simple and fast label correcting algorithm for shortest paths," *Networking*, vol. 23, no. 7, pp. 703–709, 1993.

[48] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[49] I. Chabini, "Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time," *Transportation Research Records*, vol. 1645, pp. 170–175, 1998.

[50] A. Orda and R. Rom, "Distributed shortest-path protocols for time-dependent networks," *Distributed Computing*, vol. 10, no. 1, pp. 49–62, 1996.

[51] S. Guo, Y. Gu, B. Jiang, and T. He, "Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links," in *ACM conference on Mobile computing and networking (MobiCom)*, 2009, pp. 133–144.

[52] H. Wang, X. Zhang, F. Abdesselam, and A. Khokhar, "Dps-mac: An asynchronous mac protocol for wireless sensor networks," vol. 7, June 2007, pp. 393–404.

[53] R. Zheng and R. Kravets, "On-demand power management for ad hoc networks," in *IEEE Computer and Communications Societies (Infocom)*, vol. 1, 2003, pp. 481–491.

[54] E.-Y. Lin, J. Rabaey, and A. Wolisz, "Power efficient rendezvous schemes for dense wireless sensor networks," in *IEEE International Conference on Communications (ICC)*, vol. 7, June 2004, pp. 3769–3776.

[55] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare-event detection," in *Proceedings of the 4th international symposium on Information processing in sensor networks (IPSN05)*. Piscataway, NJ, USA: IEEE Press, 2005, p. 4.

[56] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel, "Delay efficient sleep scheduling in wireless sensor networks," in *24th Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom05)*, vol. 4, March 2005, pp. 2470–2481.

[57] C. J. Colbourn, J. H. Dinitz, and D. R. Stinson, "Quorum systems constructed from combinatorial designs," *Information and Computation*, vol. 169, no. 2, pp. 160–173, 2001.

[58] J. L. M. Nuno Preguica, "Revisiting hierarchical quorum systems," in *Proceedings of the The 21st International Conference on Distributed Computing Systems (ICDCS01)*. Washington, DC, USA: IEEE Computer Society, 2001, p. 264.

[59] M. Maekawa, "A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems," *ACM Transaction on Computer System*, vol. 3, no. 2, pp. 145–159, 1985.

[60] D. Malkhi and M. Reiter, "Byzantine quorum systems," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing (STOC97)*. New York, NY, USA: ACM, 1997, pp. 569–578.

[61] D. Malkhi, M. Reiter, and R. Wright, "Probabilistic quorum systems," in *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing (PODC97)*, 1997, pp. 267–273.

[62] M. Naor and A. Wool, "The load, capacity, and availability of quorum systems," *SIAM Journal on Computing*, vol. 27, no. 2, pp. 423–447, 1998.

[63] P. Dutta and D. Culler, "Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications," in *Proceedings of the 6th ACM conference on Embedded network sensor systems (Sensys 08)*, 2008, pp. 71–84.

[64] S. H. Wu, C. M. Chen, and M. S. Chen, "An asymmetric quorum-based power saving protocol for clustered ad hoc networks," in *Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS)*, 2007.

[65] Y. Wang, C. Wan, M. Martonosi, and L. Peh, "Transport layer approaches for improving idle energy in challenged sensor networks," in *Proceedings of the 2006 SIGCOMM workshop on Challenged networks (SIGCOMM Workshops)*, 2006, pp. 253 – 260.

[66] H. S. Z. I. Niven and H. L. Mongomery, *Introduction to the Theory of Numbers*. John Wiley & Sons, 1991.

[67] Y. Chen and Y. Lin, "C-mac: An energy-efficient mac scheme using chinese-remainder-theorem for wireless sensor networks," in *IEEE International Conference on Communications (ICC 07)*, June 2007, pp. 3576–3581.

[68] Y. Kuo and C. Chen, "Crt-mac: A power-saving multicast protocol in the asynchronous ad hoc networks," in *IEEE International Conference on Sensor Networks, Ubiquitous and Trustworthy Computing (SUTC)*, 2008, pp. 332 – 337.

[69] S. Biswas and R. Morris, "Exor: opportunistic multi-hop routing for wireless networks," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 133–144, 2005.

[70] D. Kim and M. Liu, "Optimal stochastic routing in low duty-cycled wireless sensor networks," in *Proceedings of the 4th Annual International Conference on Wireless Internet (WICON)*, 2008, pp. 1–9.

[71] Z. Zhong and S. Nelakuditi, "On the efficacy of opportunistic routing," june 2007, pp. 441 –450.

[72] R. C. Shah, S. Wietholter, A. Wolisz, and J. M. Rabaey, "When does opportunistic routing make sense?" in *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 350–356.

[73] K. L. Cooke and E. Halsey, "The shortest route through a network with time-dependent internodal transit times," *J. Math. Anal. Appl.*, vol. 14, pp. 493–498, 1966.

[74] B. Ding, J. X. Yu, and L. Qin, "Finding time-dependent shortest paths over large graphs," in *Proceedings of Extending database technology (EDBT'08)*, 2008, pp. 205–216.

[75] H. Chon, D. Agrawa, and A. Abbadi., "Fates: Finding a time dependent shortest path," *Mobile Data Management*, vol. 2574, pp. 165–180, 2003.

[76] G. D'Angelo, S. Cicerone, G. Di Stefano, and D. Frigioni, "Partially dynamic concurrent update of distributed shortest paths," in *International Conference on Computing: Theory and Applications*, 2007, pp. 32–38.

[77] S. C., G. D., D. F., and U. N., "A fully dynamic algorithm for distributed shortest paths," *Theoretical Computer Science*, vol. 297, no. 1-3, pp. 83–102, 2003.

[78] K. V. Ramarao and S. Venkatesan, "On finding and updating shortest paths distributively," *J. Algorithms*, vol. 13, no. 2, pp. 235–257, 1992.

[79] S. Haldar, "An "all pairs shortest paths" distributed algorithm using 2n2 messages," *J. Algorithms*, vol. 24, no. 1, pp. 20–36, 1997.

[80] G. F. Italiano, "Distributed algorithms for updating shortest paths (extended abstract)," in *Proceedings of the 5th International Workshop on Distributed Algorithms (WDAG)*. London, UK: Springer-Verlag, 1992, pp. 200–211.

[81] B. Awerbuch, I. Cidon, and S. Kutten, "Communication-optimal maintenance of replicated information," in *Proceedings of the 31st Annual Symposium on Foundations of Computer Science (SFCS)*. Washington, DC, USA: IEEE Computer Society, 1990, pp. 492–502 vol.2.

[82] P. Kyasanur, R. R. Choudhury, and I. Gupta, "Smart gossip: An adaptive gossip-based broadcasting service for sensor networks," in *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, Oct 2006, pp. 91–100.

[83] F. Wang and J. Liu, "Duty-cycle-aware broadcast in wireless sensor networks," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2009, pp. 468–476.

[84] ——, "Rbs: A reliable broadcast service for large-scale low duty-cycled wireless sensor networks," in *IEEE International Conference on Communications (ICC)*, May 2008, pp. 2416–2420.

[85] Y. Sun, O. Gurewitz, and D. B. Johnson, "Ri-mac: a receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks," in *Proceedings of the ACM conference on Embedded network sensor systems (Sensys)*, 2008, pp. 1–14.

[86] I. C. Society, "Ieee standard 802.15.4a-2007," August 31 2007.

[87] IEEE, "Ieee 802.15 $wpan^{TM}$ task group 4 (tg4)," http://www.ieee802.org/15/pub/TG4.html/.

[88] R. L. R. Thomas H. Cormen, Charles E. Leiserson and C. Stein, *Introduction to Algorithms, Second Edition*. MIT Press and McGraw-Hill., 2001.

[89] J. Blum, M. Ding, A. Thaeler, and X. Cheng, "Connected dominating set in sensor networks and manets," *Handbook of Combinatorial Optimization (2005)*, pp. 329 – 369, 2005.

[90] R. Bellman, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[91] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems (Sensys)*, 2003, pp. 14–27.

[92] I. Demirkol, F. Alagoz, H. Delic, and C. Ersoy, "Wireless sensor networks for intrusion detection: packet traffic modeling," *Communications Letters, IEEE*, vol. 10, no. 1, pp. 22–24, Jan 2006.

[93] TinyOS, "Tinyos community forum," http://www.tinyos.net/.

[94] R. Misra and C. Mandal, "Rotation of cds via connected domatic partition in ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 4, pp. 488–499, 2009.

[95] C. Schurgers and M. Srivastava, "Energy efficient routing in wireless sensor networks," in *Military Communications Conference (MILCOM 2001)*, vol. 1, 2001, pp. 357–361.

[96] A. Orda and R. Rom, "Minimum weight paths in time-dependent networks," *Networks*, vol. 21, pp. 295–319, 1991.

[97] B. Awerbuch, "Complexity of network synchronization," *Journal of the ACM (JACM)*, vol. 32, no. 4, pp. 804–823, 1985.

[98] K. M. Chandy and J. Misra, "Distributed computation on graphs: shortest path algorithms," *Communications of the ACM*, vol. 25, no. 11, 1982.

[99] A. Segall, "Distributed network protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 1, pp. 23–35, Jan 1983.

[100] J. J. Garcia-Lunes-Aceves, "Loop-free routing using diffusing computations," *IEEE/ACM Trans. Netw.*, no. 1, pp. 130–141, 1993.

[101] S. Mallat, *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, 2008.

[102] OMNET++, http://www.omnetpp.org/.

[103] M. Zuniga and B. Krishnamachari, "Analyzing the transitional region in low power wireless links," in *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2004, pp. 517–526.

[104] C. E. Perkins and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers,” *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 4, pp. 234–244, 1994.

[105] F. Stann, J. Heidemann, R. Shroff, and M. Z. Murtaza, “Rbp: robust broadcast propagation in wireless networks,” in *ACM SenSys*, 2006, pp. 85–98.

[106] J. Blum, M. Ding, A. Thaeler, and X. Cheng, “Connected dominating set in sensor networks and manets,” *Handbook of Combinatorial Optimization*, pp. 329–369, 2005.

[107] A. Qayyum, L. Viennot, and A. Laouiti, “Multipoint relaying for flooding broadcast messages in mobile wireless networks,” in *35th Annual Hawaii International Conference on System Science*, 2002, pp. 3866–3875.

[108] L. Lamport, “Time,clocks, and the ordering of events in a distributed system,” *Communications of the ACM*, vol. 21, 1978.